TRANSDUCTIVE SEMANTIC PARSING

by

Sheng Zhang

A dissertation submitted to Johns Hopkins University in conformity with the requirements for the degree of Doctor of Philosophy

Baltimore, Maryland

February 2020

Abstract

Semantic parsing aims at mapping natural language text into meaning representations, which have the potential to facilitate semantic analysis, and more importantly, to transform how humans interact with machines. While semantic parsing receives a long-standing interest from the community, developing robust semantic parsing algorithms remains a challenging problem. In this thesis, we consider several challenges in semantic parsing: 1) representing the semantics of multiple natural languages in a single semantic analysis; 2) developing parsing systems for broad-coverage semantics; 3) designing unifying parsing paradigms to support distinct meaning representation frameworks; and 4) training systems with limited amounts of labeled data.

We approach semantic parsing as sequence-to-graph transduction problems, and introduce novel algorithms/components into transductive settings that extend beyond what a typical neural machine translation system would do on this problem. Our approach achieves the state-of-the-art performance on a number of tasks, including cross-lingual open information extraction, cross-lingual decompositional semantic parsing, and broad-coverage semantic parsing for Abstract Meaning Representation (AMR), Semantic Dependencies (SDP) and Universal Conceptual Cognitive Annotation (UCCA).

In the first half of this thesis, we are concerned with representing the semantics of multiple natural language in a single semantic analysis. We introduce two cross-lingual semantic processing tasks: cross-lingual information extraction and cross-lingual decompositional semantic parsing. We propose end-to-end sequence transduction models, and present an evaluation metric that can be used to differentiate two meaning representations with similar instances, analysis, or attributes. Experiments show that our approach significantly outperforms strong baselines, and extension to low-resource scenarios also gains promising improvement.

In the second half, we focus on developing parsing systems that support broad-coverage meaning representation frameworks with rich graph-based semantic formalism. We unify different broad-coverage semantic parsing tasks under a transduction paradigm, and propose attention-based neural models that build a meaning representation via sequence-to-graph transduction. Experiments conducted on three separate broad-coverage semantic parsing tasks – AMR, SDP and UCCA – demonstrate that our attention-based neural transducer improves the state of the art on both AMR and UCCA, and is competitive with the state of the art on SDP.

Finally, we conclude the thesis, and outline ideas and suggestions for future directions of transductive semantic parsing.

Primary Readers and Advisors: Benjamin Van Durme and Kevin Duh

Secondary Reader: Kyle Rawlins

Acknowledgments

This past four years at Johns Hopkins have been an unforgettable and life-changing experience to me. I would not have been able to make this journey without the help and support of many people.

First and foremost, I am extremely fortunate to have Benjamin Van Durme and Kevin Duh as my advisors. I would like to thank Ben for his guidance, advice and teaching throughout my studies. I knew little about semantics and common sense when I first came to Johns Hopkins. Ben has taught me a lot in the field through his sparking, high-level views as well as his patient and detail-oriented explanations in our every meeting. I am inspired by his erudition and insight, his curiosity and energy, and his humor and optimism. I have been honored to have Kevin as my secondary advisor soon after he joined CLSP. Kevin is a kind, caring and supportive advisor. He is always invigorating and generous to share practical suggestions and hands-on experiences in our every discussion. Both Ben and Kevin gave me a lot of freedom to explore my research directions, while being warm and helpful whenever I was in need of guidance. I have benefited greatly from their unlimited support and advice on every aspect of my studies and future career. Thank you, Ben and Kevin!

I would like to thank Kyle Rawlins for being on my thesis committee and for his help throughout my studies. His course of semantics broadened my knowledge and understanding of what "*meaning*" means, and influenced my research. It was also my great honor to have Aaron Steven White and Philipp Koehn on my graduate board oral committee. Their thought-provoking comments on the transductive framework and meaning representations helped shape my direction in the part of my thesis study.

JHU-CLSP offers me a fantastic environment for learning as well as having fun. I have learned a lot from many faculties here, especially Raman Arora, Matt Post, Jason Eisner, Mark Dredze, Shinji Watanabe, Sanjeev Khudanpur and Hynek Hermansky. I am also lucky to be surrounded by my awesome CLSP present and former colleagues: Rachel Rudinger, Xutai Ma, Tim Vieira, Keisuke Sakaguchi, Pushpendre Rastogi, Adam Poliak, Chandler May, Adam Teichert, Nanyun Peng, Hongyuan Mei, Shuoyang Ding, Dingquan Wang, Tongfei Chen, Nanxin Chen, Ke Li, Chunxi Liu, Ruizhe Huang, Xiaohui Zhang, Guanghui Qin, Yiming Wang, Patrick Xia, Chu-Cheng Lin, Ting Hua, Elias Stengel-Eskin, Ryan Culkin, Seth Ebner, Nils Holzenberger, Mahsa Yarmohammadi, Kenton Murray, Huda Khayrallah, Dee Ann Reisinger, Adrian Benton, Lisa Bauer, Zachary Wood-Doughty, Sabrina Mielke, Xuchen Yao, Guoguo Chen, Travis Wolfe, Francis Ferraro, Nicholas Andrews, Matt Gormley, Juri Ganitkevitch, Craig Harman, Tom Lippincott and certainly others.

During my PhD, I have two enjoyable internships at Microsoft Research and Amazon Alexa AI. I benefited tremendously from my mentors Xiaodong Liu, Jingjing Liu, Jianfeng Gao, Arpit Gupta and Lambert Mathias. I am also thankful for wonderful people I met during the internships: Shuohang Wang, Lianhui Qin, Liu Yang, Yichong Xu, Yuexin Wu, Huazheng Wang, Junjie Hu, Kun Yuan, Yitong Li, Shang-Yu Su, Robin Jia, Yelong Shen, Zhe Gan, Yu Cheng, Hancheng Ge, Hua He and Chetan Naik.

During my job seeking season, I visited several great places where many people have warmly hosted me and gave me valuable advice. I wish to thank for their great help: Peter Clark, Hoifung Poon, Yashar Mehdad, Oren Etzioni, Ves Stoyanov, Kristina Toutanova and Luke Zettlemoyer.

A special thanks to Yansong Feng for the life-changing experience with him while I was a master student at Peking University. My PhD studies would have not been possible without his mentorship and support. I also sincerely thank my wonderful fellows at Wangxuan Institute of Computer Technology of Peking University: Kun Xu, Zhe Han, Liwei Chen, Bingfeng Luo, Lei Zhou and Dongyan Zhao.

This degree would not have been possible without the steadfast support of my family. I would like to thank my wife, Xinyi Zhong, for her unconditional love, personal sacrifice and unfair share of family responsibilities. Whenever I am depressed, she is always there to cheer me up. I am privileged to have her with me in this journey. I am indebted to my parents, Yiping Zhang and Hongying Wei. I can always feel their endless love and support traveling across the Pacific. They made me who I am, and I never know how to pay them back. I also wish to thank my little corgi, Kero, who loves to chew his toys beside me and brings tremendous joy to my life.

To Xinyi

Table of Contents

List of Tables

List of Figures

Chapter 1

Introduction

1.1 Motivation

As the primary medium for us to understand and talk about the world, language is central to human cognition and communication. The sophistication language offers us to express the world has sparked linguistics' interest in representing the meaning of human language. Meanwhile, computers do not genuinely understand human language, which prevents machines from conversing naturally with humans. This barrier separating humans from machines attracts considerable attention from the community of computational linguistics, where systems have being built for mapping natural language text into logical forms or meaning representations – semantic parsing. The past decades have witnessed improvements of semantic parsing systems from a set of hand-coding rules in the early days (Green Jr et al., 1961; Woods et al., 1972; Winograd, 1972) to statistical learning methods based on hand-crafted features (Zelle and Mooney, 1996; Zettlemoyer and Collins, 2005; Poon and Domingos, 2009; Kwiatkowski et al., 2011; Liang et al., 2013; Berant et al., 2013; Artzi et al., 2015; inter alia) to today's deep learning methods (Dong and Lapata, 2016; Jia and Liang, 2016; Ling et al., 2016; Yin and Neubig, 2017; Liang et al., 2017; Zhang et al., 2019a; inter alia).

The levels of accuracy achieved by these semantic parsing systems have led to their use in semantic analysis (e.g. Toutanova et al., 2002; Bos et al., 2004), and more importantly, natural language interfaces to machines (e.g. Kwiatkowski et al., 2011; Dong and Lapata, 2016). However, the usefulness of these systems is mostly limited to specific domains, such as ATIS (Bates et al., 1990; Price, 1990), GEO and JOBS (Zettlemoyer and Collins, 2005), FREE917 (Cai and Yates, 2013), WEBQUESTIONS (Berant et al., 2013), IFTTT (Quirk et al., 2015), etc. In contrast, natural language affords a much broader coverage in terms of domains, genres, and even the language itself. This gap of coverage brings renewed interest in designing meaning representations for a wide range of natural languages. Over the past years, a number of broad-coverage meaning representation frameworks have been proposed and annotated, including Abstract Meaning Representation (AMR; Banarescu et al., 2013), Universal Conceptual Cognitive Annotation (UCCA; Abend and Rappoport, 2013), Semantic Dependency Parsing (SDP; Oepen et al., 2014; Oepen et al., 2015), and Universal Decompositional Semantics (UDS; White et al., 2016).[1] These general-purpose meaning representations introduce new challenges in developing semantic parsing systems:

Lexical Mismatch There is an increasing number of efforts on cross-lingual

[1]There are further calls to attend to existing efforts as well, e.g., Episodic Logic (Schubert, 2000; Hwang and Schubert, 1994; Schubert and Hwang, 2000; Schubert, 2014), or Discourse Representation Theory (Kamp, 1981; Heim, 1988).

semantic analysis, i.e., representing the semantics of multiple natural languages in a single semantic analysis (Yarowsky et al., 2001; Padó and Lapata, 2009; Evang and Bos, 2016; Abzianidze et al., 2017), which aims at enabling semantic analysis on non-English languages. A challenging issue along with this goal is the lexical mismatch between the source text and the target meaning representations that are usually based on English. Under these cross-lingual settings, the vocabulary sizes on both sides are much larger than those in domain-specific semantic parsing, and they have little overlap with each other, which poses a challenge in building robust cross-lingual semantic parsing systems. A similar issue also arises in some monolingual settings, e.g. AMR parsing (Banarescu et al., 2013), where the meaning representation appears to "abstract" from the syntactic realizations, leaving no explicit correspondence between elements of the meaning representation and the surface utterance.

Structural Complexity While parsing has long been dominated by tree-structured representations, most broad-coverage semantic parsing targets graph-structured representations. On one hand, graph-structured representations are more expressive and arguably more adequate for sentence-level analysis beyond surface syntax and in particular for the representation of semantic structure. For instance, reentrancy – the same semantic concept can participate in multiple relations – requires multiple incoming edges to a node in the semantic structure, which is beyond what a traditional tree structure can represent. On the other hand, being structurally more complex than trees, graph-structured representations exhibit higher degrees of non-projectivity, reentrancies, and partial connectivity (Oepen et al., 2014; Damonte et al., 2017). This structural complexity calls for more general graph-oriented parsing systems or extensions to traditional tree-oriented parsing systems.

Framework Balkanization Renewed interest in broad-coverage semantic analysis has led to a surge of proposed new frameworks, e.g., AMR, UCCA, SDP, and UDS. However, these new meaning representation frameworks are balkanized, i.e., they have different *formal* and *linguistic* assumptions. As a consequence, a variety of framework-specific parsing approaches has been developed over the past years. For instance, the state-of-the-art SDP parsers (Dozat and Manning, 2018; Peng et al., 2017a) are not directly transferable to AMR and UCCA because of the lack of explicit correspondence between words in the sentence and nodes in the semantic graph. Reducing framework-specific balkanization or developing a unifying parsing approach have hardly been explored (with notable exceptions, e.g. Peng et al., 2017a; Hershcovich et al., 2018; Stanovsky and Dagan, 2018; Oepen et al., 2019).

Limited Data While general-purpose meaning representations allow for broader coverage and rich semantic information, annotating training data for these representations is time consuming and requires expertise. The expensiveness of data annotation results in relatively limited amounts of labeled data. For instance, the most recent official AMR corpus provides no more than 40,000 training instances. Such a small amount of training data becomes a bottleneck in boosting the performance of semantic parsers. Recently several solutions have been used to alleviate data sparsity, including linearization (Peng et al., 2017b) and data augmentation (Konstas et al., 2017; Noord and Bos, 2017b), but their use of neural network architectures is still limited.

1.2 Thesis Outline

Following the challenges we just discussed, in Chapter 2 we first provide summary background of meaning representation frameworks targeted in this thesis and review related work on parsing for each; we then present PredPatt – a pattern-based framework for predicate-argument extraction (Zhang et al., 2017e). PredPatt automatically creates the skeleton of Universal Decompositional Semantics (UDS; White et al., 2016) from raw sentences.

After the background chapter, the rest of this thesis consists of two parts – PART I CROSS-LINGUAL SEMANTIC PARSING and PART II BROAD-COVERAGE SEMANTIC PARSING.

PART I focuses on semantic parsing in cross-lingual settings. We start with representing predicate-argument structures of multiple natural languages in a single shallow semantic analysis, and then move to a form of decompositional analysis, which is designed to allow systems to target varying levels of structural complexity.

In Chapter 3, we present a series of solutions to cross-lingual open information extraction. We first introduce the problem cross-lingual open IE: distilling facts from foreign language into shallow semantic representations in another language. We then propose a joint solution based on a neural sequence-to-sequence model. Next, we improve our approach via a novel selective decoding mechanism, and a simple and effective training technique *Halo*. Experimental results show that our approaches achieve consistent and significant improvements over strong baselines in a variety of cross-lingual open IE scenarios. This chapter is based on our work in Zhang et al. (2017b); Zhang et al. (2017c); Mei et al. (2018).

In Chapter 4, we introduce the task of cross-lingual decompositional semantic parsing, which maps the content provided in the source language into decompositional analysis based on the target language. We present: UDS graph/linearized representations as the target semantic interface, a new evaluation metric, and a Chinese-English decompositional semantic parsing dataset. We propose an end-to-end learning approach with a coreference annotating mechanism, surpassing strong baselines. We separately evaluate the coreference mechanism and Semantic Proto-Role prediction, showing promising results. This chapter is based on our work in Zhang et al. (2018).

PART II studies semantic parsing in monolingual settings, where we are concerned with representing rich semantics of a wide range of monolingual text, i.e., broad-coverage semantic parsing. We recast it as a sequence-to-graph transduction problem, and propose a series of novel components into a typical neural sequence transduction system.

In Chapter 5, we propose an attention-based model that treats AMR parsing as sequence-to-graph transduction. Unlike most AMR parsers that rely on pre-trained aligners, external semantic resources, or data augmentation, our proposed parser is aligner-free, and it can be effectively trained with limited amounts of labeled AMR data. Our experimental results outperform all previously reported SMATCH F1 scores. We provide thorough analysis of contributions made by each component in our model. This chapter is based on our work in Zhang et al. (2019a).

In Chapter 6, we unify different broad-coverage semantic parsing tasks under a transduction paradigm, and propose an attention-based neural framework that incrementally builds a meaning representation via a sequence of semantic relations. Experiments conducted on three separate semantic parsing tasks – AMR, SDP and UCCA – demonstrate that our framework improves the state of the art on both AMR and UCCA, and is competitive with the state of the art on SDP. This chapter is based on our work in Zhang et al. (2019b).

1.3 Contributions

The contributions of this thesis are summarized as below:

- We introduce PredPatt, a framework of extensible, interpretable, language neutral predicate-argument extraction patterns. PredPatt bridges the deep syntax of the Universal Dependency project to an initial shallow semantic layer of Universal Decompositional Semantics.

- We pioneer the research direction of applying neural transduction approaches to cross-lingual semantic parsing. In particular, we propose variants of neural sequence-to-sequence models as core components of end-to-end solutions for cross-lingual semantic analysis that targets varying levels of structural complexity.

- We unify broad-coverage semantic parsing tasks under a transduction setting, and introduce a series of novel components into the transductive setting that extend beyond what a typical neural machine translation system would do on these tasks.

Chapter 2

Background

2.1 Meaning Representations

The ultimate goal of meaning representations is to represent the complete meaning of natural language text in a fully formal language that (1) has a rich ontology of types, properties, and relations; and (2) supports automated reasoning or execution. There is a long and rich history of meaning representations (Mooney, 2014), including several milestones as follow: Gottfried Leibniz in 1685 developed a formal conceptual language, the *characteristica universalis*, for use by an automated reasoner, the *calculus ratiocinator*. Whitehead and Russell (1912) finalized the development of modern first-order predicate logic that gave the logical form of human reasoning. Montague (1970) proposed the marriage between lambda calculus and syntax, and developed the fundamental principle of *semantic compositionality*. Accompanying with the development of meaning representations, we have seen (1) the rise and fall of manually developed semantic interfaces for databases (Green Jr et al., 1961; Woods et al., 1972; Winograd, 1972); (2) the renaissance of semantic parsing whose foci were automatic learning with statistical methods (Zelle and Mooney, 1996; Tang and Mooney, 2001; Zettlemoyer and Collins, 2005), reducing the supervision form from full meaning representations to weak supervision or unsupervised learning (Clarke et al., 2010; Berant et al., 2013; Poon and Domingos, 2009), and grounded learning that connected the use of language to specific domains (MacMahon et al., 2006; Chen and Mooney, 2008; Artzi and Zettlemoyer, 2013); (3) the blossom of recent work on neural semantic parsing that requires less hand-crafted feature engineering, but is much more data-hungry (Dong and Lapata, 2016; Jia and Liang, 2016; Ling et al., 2016). In response to the trend of semantic parsing, there is increasing interest in annotating more data with meaning representations that are designed for a wide range of natural language text, and have broader coverage in terms of semantic phenomena (Banarescu et al., 2013; Oepen et al., 2014; Abend and Rappoport, 2013; White et al., 2016; White et al., 2019). These meaning representation annotations are expected to lead to new work in natural language understanding, resulting in semantic parsers that are as ubiquitous as syntactic ones, and supporting natural language generation by providing a logical semantic input. In section, we provide summary background on the meaning representations we target, and review the related work on parsing for each.

2.1.1 Abstract Meaning Representation

Abstract Meaning Representation (AMR; Banarescu et al., 2013) encodes sentence-level semantics, such as predicate-argument information, reentrancies, named entities, negation and modality, into a rooted, directed, and usually acyclic graph with node and edge labels. AMR graphs abstract away from syntactic realizations. For example, the sentences “he described her as a genius”, “his description of her: genius”, and “she was a genius, according to his description” are all assigned the same AMR. AMR makes extensive use of PropBank framesets (Palmer et al., 2005). For example, AMR represents a phrase like “bond investor” using the frame “invest-01”, even though no verbs appear in the phrase. AMR is heavily biased towards English. It is not an Interlingua. Figure 2.1 shows the AMR graph of “The boy wanted to go", which is equivalent to the following logic format:[1]

$$\exists w, b, g : \text{instance}(w, \text{want-01}) \wedge \text{instance}(g, \text{go-01}) \wedge \text{instance}(b, \text{boy})$$

$$\wedge\, \text{arg0}(w, b) \wedge \text{arg1}(w, g) \wedge \text{arg0}(g, b)$$

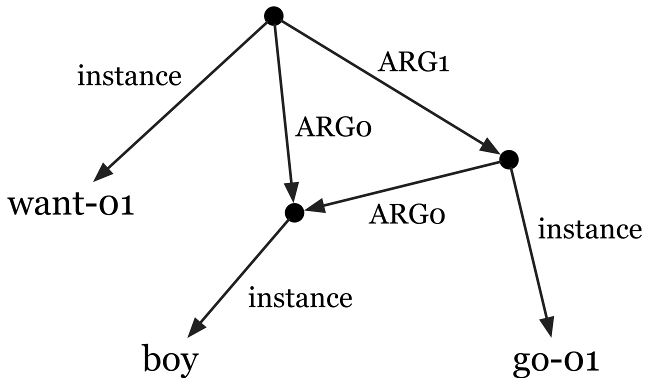


Figure 2.1: The AMR graph of “The boy wanted to go".

Since its first general release in 2014, AMR has been a popular target of data-driven semantic parsing, notably in two SemEval shared tasks (May, 2016; May and Priyadarshi, 2017). Graph-based parsers build AMRs by identifying

[1]See Banarescu et al. (2013) for detail.

concepts and scoring edges between them, either in a pipeline (Flanigan et al., 2014), or jointly (Zhou et al., 2016; Lyu and Titov, 2018). This two-stage parsing process limits the parser incrementality. Transition-based parsers either transform dependency trees into AMRs (Wang et al., 2015; Wang et al., 2016; Goodman et al., 2016), or employ transition systems specifically tailored to AMR parsing (Damonte et al., 2017; Ballesteros and Al-Onaizan, 2017). Transition-based parsers rely on the pre-trained aligner to produce the reference transitions. Grammar-based parsers leverage external semantic resources to derive AMRs compositionally based on CCG rules (Artzi et al., 2015), or SHRG rules (Peng et al., 2015). Another line of work uses neural model translation models to convert sentences into *linearized* AMRs (Barzdins and Gosko, 2016a; Peng et al., 2017b), but has to rely on data augmentation to produce effective parsers (Noord and Bos, 2017b; Konstas et al., 2017). Our approaches in this thesis differ from them in that they do not rely on pre-trained aligners, and can be effectively trained without data augmentation.

2.1.2 Semantic Dependency Parsing

Semantic Dependency Parsing (SDP) was introduced in 2014 and 2015 SemEval shared tasks (Oepen et al., 2014; Oepen et al., 2015). It is centered around three semantic formalisms – DM (DELPH-IN MRS; Flickinger et al., 2012; Oepen and Lønning, 2006), PAS (Predicate-Argument Structures; Miyao and Tsujii, 2004), and PSD (Prague Semantic Dependencies; Hajič et al., 2012) – representing predicate-argument relations between content words in a sentence. Their annotations have been converted into bi-lexical dependencies, forming directed graphs whose nodes injectively correspond to surface lexical units, and edges represent semantic relations between nodes. Figure 2.2 shows an example DM graph. It can be characterized as a labeled, directed graph $G = (V, E, \ell_V, \ell_E)$ where $V = \{1, ..., n\}$ is a set of *nodes* (which are in one-to-one correspondence with the tokens of the sentence $x = x_1, ..., x_n$); $E \subseteq V \times V$ is a set of *edges*; and ℓ_V and ℓ_E are mappings that assign *labels* (from some finite alphabet) to nodes and edges, respectively. See Oepen et al. (2014) for detail.

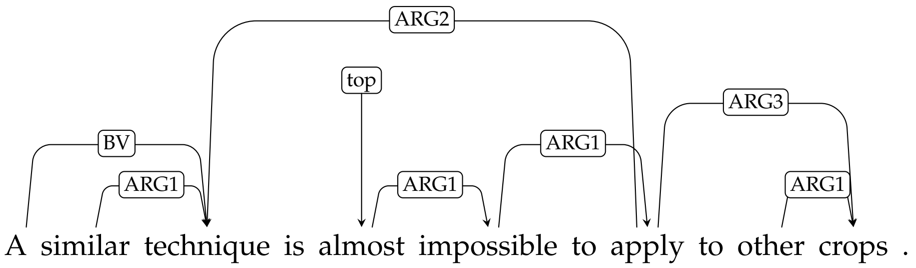

ARG2
top
ARG3
BV
ARG1
ARG1
ARG1
ARG1
A similar technique is almost impossible to apply to other crops .

Figure 2.2: An example SDP:DM graph.

Most recent parsers for SDP are graph-based: Peng et al. (2017a); Peng et al. (2018) use a max-margin classifier on top of a BiLSTM, with the factored score for each graph over predicates, unlabeled arcs, and arc labels. Multi-task learning approaches and disjoint data have been used to improve the parsing performance. Dozat and Manning (2018) extend an LSTM-based syntactic dependency parser to produce graph-structured dependencies, and carefully tune it to the state of the art performance. Wang et al. (2018) extend the transition system of Choi and McCallum (2013) to produce non-projective trees, and use improved versions of stack-LSTMs (Dyer et al., 2015) to learn the representation for key components. All of these are specialized for bi-lexical dependency parsing, whereas our solutions in this thesis are general enough to produce both bi-lexical semantics graphs and other types of semantic graphs that are less anchored to the surface utterance.

2.1.3 Universal Conceptual Cognitive Annotation

Universal Conceptual Cognitive Annotation (UCCA; Abend and Rappoport, 2013) targets a level of semantic granularity that abstracts away from syntactic paraphrases in a typologically-motivated, cross-linguistic fashion. Sentence representations in UCCA are directed acyclic graphs (DAG), where terminal nodes correspond to surface lexical tokens, and non-terminal nodes to semantic units that participate in super-ordinate relations. Edges are labeled, indicating the role of a child in the relation the parent represents. Figure 2.3 shows an example UCCA DAG, representing the UCCA foundational layer. The foundational layer views the text as a collection of *Scenes*. A Scene can describe some movement or action, or a temporally persistent state. Table 2.1 shows the categories used by the example UCCA DAG in Figure 2.3. See Abend and Rappoport (2013) for detail.

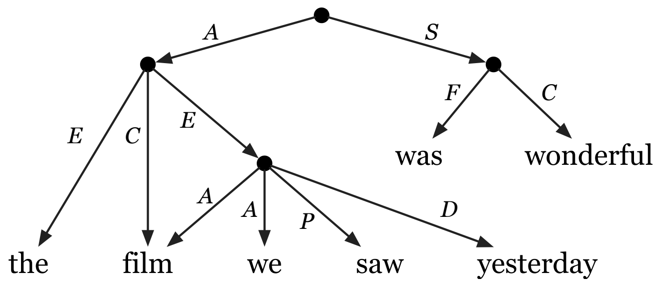


Figure 2.3: An example UCCA graph.

The first UCCA parser is proposed by Hershcovich et al. (2017), where

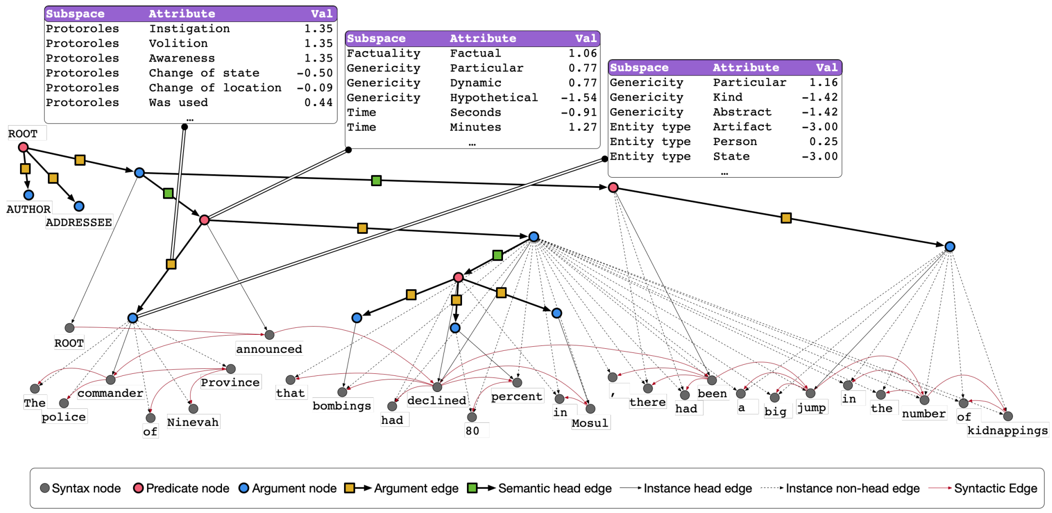
Abb.	Category	Short Definition
P	Process	The main relation of a Scene that evolves in time (usually an action or movement).
S	State	The main relation of a Scene that does not evolve in time
A	Participant	A participant in a Scene in a broad sense (including locations, abstract entities and Scenes serving as arguments).
D	Adverbial	A secondary relation in a Scene (including temporal relations).
C	Center	Necessary for the conceptualization of the parent unit
E	Elaborator	A non-Scene relation which applies to a single Center
F	Function	Does not introduce a relation or participant. Required by the structural pattern it appears in.

Table 2.1: Categories used by the example UCCA graph.

they extend a transition system to produce DAGs. To leverage other semantic resources, Hershcovich et al. (2018) is one of the few attempts to present (lossy) conversion from AMR, SDP and Universal Dependencies (UD; Nivre et al., 2016) to a unified UCCA-based DAG format. They explore multi-task learning under the unified format. While multi-task learning improves UCCA parsing results, it shows poor performance on AMR, SDP and UD parsing. In contrast, different semantic parsing tasks in this thesis are formalized in a unified transduction paradigm with no loss, and our approach achieves the state-of-the-art or competitive performance on each task.

2.1.4 Universal Decompositional Semantics

Universal Decompositional Semantics (UDS; Reisinger et al., 2015; White et al., 2016) addresses the brittleness of category-based systems in traditional



Subspace	Attribute	Val
Protoroles	Instigation	1.35
Protoroles	Volition	1.35
Protoroles	Awareness	1.35
Protoroles	Change of state	-0.50
Protoroles	Change of location	-0.09
Protoroles	Was used	0.44
	…	

Subspace	Attribute	Val
Factuality	Factual	1.06
Genericity	Particular	0.77
Genericity	Dynamic	0.77
Genericity	Hypothetical	-1.54
Time	Seconds	-0.91
Time	Minutes	1.27
	…	

Subspace	Attribute	Val
Genericity	Particular	1.16
Genericity	Kind	-1.42
Genericity	Abstract	-1.42
Entity type	Artifact	-3.00
Entity type	Person	0.25
Entity type	State	-3.00
	…	

Figure 2.4: An example Universal Decompositional Semantics graph.

semantic annotation frameworks by decomposing complex and often exclusive categories into a set of semantic features (Dowty, 1989). Annotations of these features take the form of many simple questions about words or phrases (in context) that are easy to naïve native speakers to answer, thus allowing annotations to be crowd-sourced while retaining high inter-annotator agreement. Beside the ease of annotation, UDS is highly extensible – the natural of decompositonal semantics retains the ability to capture feature configurations that were not considered at design time. Reannotation after an overhaul of the framework's ontology is never necessary, since additional annotations simply accrue to sharpen the framework's ability to capture fine-grained semantic phenomena. Now a broad coverage of linguistic phenomena have been considered in the UDS dataset (White et al., 2019), including semantic roles (Reisinger et al., 2015; White et al., 2016), entity types (White et al., 2016), event factuality (White et al., 2016; Rudinger et al., 2018b), linguistic expressions of generalizations about entities and events (Govindarajan et al., 2019), and temporal properties of and relations between events (Vashishtha et al., 2019).

UDS consists of three layers of annotations: (i) syntactic graphs built from Universal Dependencies (UD); (ii) semantic graphs built from the predicate-argument structures deterministically extracted by the PredPatt tool (White et al., 2016; Zhang et al., 2017e) from UD; and (iii) semantic types for predicates, arguments, and their relationships, derived from decompositional semantics. Figure 2.4 shows an example UDS graph with all three layers of annotation. Semantic types at the third layer of annotations have been modeled separately: Rudinger et al. (2018b) employ linear-chain LSTMs and tree LSTMs to predict event factuality; Rudinger et al. (2018a) introduce neural-Davisonian framework to predict semantic proto-roles; Govindarajan et al. (2019) and Vashishtha et al. (2019) leverage pre-trained encoders for linguistic expressions of generalization and temporal relation prediction respectively. Recently Stengel-Eskin et al. (2019) present a sequence-to-graph transductive model for joint UDS parsing, which learns to extract both UDS graph structures and attributes from natural language input. Comparing against a strong pipeline system, the transductive parser performs comparably to the pipeline while additionally learning to produce decompositional attribute scores.

2.2 PredPatt

In this section, we introduce PredPatt[2], a pattern-based framework for predicate-argument extraction. PredPatt defines a set of interpretable, extensible and non-lexicalized patterns based on Universal Dependencies (UD, Marneffe

[2]PredPatt is publicly available at `https://github.com/hltcoe/PredPatt`

et al., 2014), and extracts predicates and arguments through these manual patterns. Figure 2.5 shows the predicates and arguments extracted by PredPatt from the sentence: "*Chris, the designer, wants to launch a new brand.*"

(1) [Chris, the designer] **wants** [to launch a new brand]
(2) [Chris, the designer] **to launch** [a new brand]
(3) [Chris] **be** [the designer]

Figure 2.5: Predicates and arguments extracted by PredPatt.[3]

The underlying predicate-argument structure constructed by PredPatt is a directed graph, where a special dependency `ARG` is built between a predicate head token and its arguments' head tokens, and the original UD relations are retained within predicate phrases and argument phrases. For example, Figure 2.6 shows the directed graph for the predicate-argument extraction (1) and (2) in Figure 2.5: The predicates are colored blue in dotted cycles with gray background. The arguments are colored purple in solid cycles. The head tokens of predicates and arguments are underlined in bold. A special dependency `ARG` is built between a predicate head token and its arguments head tokens. The UD relations are kept within predicates and arguments. The relations between predicate head tokens are also kept. The upper relations are UD. The lower relations are `ARG` relations added by PredPatt.

Compared to other existing systems for predicate-argument extraction (Banko et al., 2007; Fader et al., 2011; Angeli et al., 2015), the use of manual language-agnostic patterns on UD makes PredPatt a well-founded component

[3]The predicates are colored blue, and the arguments are colored purple with brackets.

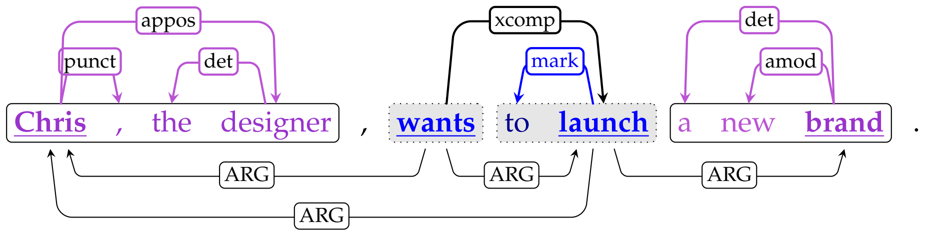


Figure 2.6: Underlying predicate-argument structure constructed by PredPatt.

across languages. Additionally, the underlying structure constructed by PredPatt has been shown to be a well-formed syntax-semantics interface for NLP tasks: Zhang et al. (2017d) utilizes PredPatt to extract possibilistic propositions in automatic common-sense inference generation. White et al. (2016) uses PredPatt to help augmenting data with *Universal Decompositional Semantics*. In this thesis, PredPatt used to generate training data for cross-lingual semantic parsing.

However, at the time of this work, the evaluation of PredPatt had been restricted to manually-checked extractions over a small set of sentences (White et al., 2016), which lacked gold annotations to conduct an objective and reproducible evaluation, and inhibited the updates of patterns in PredPatt.

In this section, we aim to conduct a large-scale and reproducible evaluation of PredPatt by introducing a large set of gold annotations gathered from PropBank (Palmer et al., 2005). We leverage these gold annotations to improve PredPatt and compare it with other prominent systems. The evaluation results demonstrate that we make a promising improvement on PredPatt, and it significantly outperforms other comparing systems.

2.2.1 Creating Gold Annotations

Open Information Extraction (Open IE) and Semantic Role Labeling (SRL) (Carreras and Màrquez, 2005) are quite related: semantically labeled arguments correspond to the arguments in Open IE extractions, and verbs often match up with Open IE relations (Christensen et al., 2011). Lang and Lapata (2010) has acknowledged that the SRL task can be viewed as a two stage process of (1) recognizing predicates and arguments then (2) assigning semantics. Therefore, predicate-argument extraction (i.e., Open IE) should primarily be considered the same as the first of two stages of SRL, and expert annotated SRL data would be an ideal resource for evaluating Open IE systems. This makes PropBank (Palmer et al., 2005) a natural choice from which we can create gold annotations for Open IE. Here, we choose to use expert annotations from PropBank, as compared to the recent suggestion to employ non-expert annotations as a means of benchmarking systems (Stanovsky and Dagan, 2016). Another advantage of choosing PropBank is that PropBank has gold annotations for UD which lays the important groundwork for evaluating UD-based patterns in PredPatt.

We create gold annotations for predicate-argument extraction by converting PropBank annotations on English Web Treebank (EWT) (LDC2012T13) and the Penn Treebank II Wall Street Journal Corpus (WSJ) (Marcus et al., 1994). These two corpora have all verbal predicates annotated, and are used to evaluate PredPatt in different perspectives: EWT is the corpus where the gold standard English UD Treebank is built over, which enables an evaluation and analysis of PredPatt patterns; WSJ is used to evaluate PredPatt in a real-world scenario where we run SyntaxNet Parser[4] (Andor et al., 2016) on the corpus to generate automated UD parses as input of PredPatt.

Table 2.2 shows the statistics of the auto-converted gold annotations for predicate-argument extraction on EWT and WSJ. We convert the PropBank annotations for all verbal predicates in these two corpora, and ignore roles of directional (DIR), manner (MNR), modals (MOD), negation (NEG) and adverbials (ADV), as they aren't extracted as distinct argument but instead are folded into the complex predicate by PredPatt and other systems for predicate-argument extraction (Banko et al., 2007; Fader et al., 2011; Angeli et al., 2015). For EWT, we select 13,583 sentences that have the version 2.0 of the gold UD annotations.[5] The resulting annotations on these two corpora contain over 94K extractions.

Corpus	#sentence	#predicate	#unique_verb	#avg_arg_per_pred
EWT	13,583	21,479	4,336	2.0
WSJ	36,432	73,076	7,880	2.1

Table 2.2: Statistics of the gold annotations on EWT and WSJ.

2.2.2 Improving PredPatt

PredPatt is a pattern-based system, comprising an extensible set of clean, interpretable linguistic patterns over UD parses. By analyzing PredPatt extractions in comparison with gold annotations (Sec. 2.2.1), we are able to refine and improve PredPatt's pattern set. From the auto-converted gold annotations, we create a held-out set by randomly sampling 10% sentences from EWT. We

[4]SyntaxNet Parser is trained on the UD Treebank which has no overlap with WSJ.
[5]English UD Treebank is available at: `http://universaldependencies.org`

then update the existing PredPatt patterns and introduce new patterns by analyzing PredPatt annotations on the held-out set.

PredPatt extracts predicates and arguments in four stages (White et al., 2016): (1) predicate and argument root identification, (2) argument resolution, (3) predicate and argument phrase extraction, and (4) optional post-processing. We analyze PredPatt extraction in each of these stages on the held-out set, and make 19 improvements to PredPatt patterns. Due to lack of space, we only highlight one improvement for each stage below.

Fixed-MWE-pred: The UD version 2.0 introduces a new dependency relation `fixed` for identifying fixed function-word "multiword expressions" (MWEs). To accommodate this new feature, we add patterns to identify the MWE predicate and its argument. As shown in Figure 2.7, the predicate root in this case is the dependent of `fixed` that is tagged as a verb (i.e., "opposed"); the root of its argument is the token which indirectly governs the predicate root via the `case` and `fixed` relation (i.e., "one").

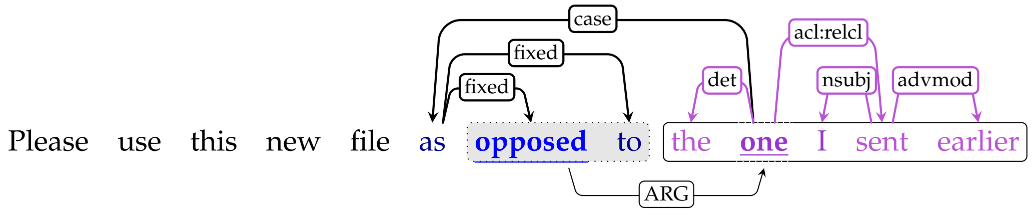

case
fixed
fixed
acl:relcl
det
nsubj
advmod
Please use this new file as opposed to the one I sent earlier
ARG

Figure 2.7: Example for add argument for `fixed` MWE predicates.

Cut-complex-pred: The existing patterns take clausal complements (`ccomp` and `xcomp`) as *predicatives* of complex predicates in the argument resolution stage, where the arguments of the clausal complement will be merged into the argument set of their head predicate. For example, in the sentence "*Chris,* *the designer, wants to launch a new brand*", PredPatt merges the argument "*a new brand*" of the predicate "*to launch*" into the argument set of the complex predicate "*wants to launch*". As a result, only the complex predicate, "[Chris, the designer] **wants to launch** [a new brand]", will be extracted. It ignores the possibility of the clausal complement itself being a predicate. Here, we add a cutting option; when turned on, it will cut the complex predicate into simple predicates as shown in Figure 2.5.

Prep-separation: By default, PredPatt considers prepositions to belong to the predicate, while PropBank places preopositions within the span of their corresponding argument. Either behavior may be preferable under different circumstances, so we make preposition placement a new configurable option of PredPatt.

Borrow-subj-for-conj-of-xcomp: PredPatt contains a post-processing option for distributing a single `nsubj` argument over multiple predicates joined by a `conj` relation. PredPatt also contains a pattern assigning subject arguments to predicates introduced by open clausal complement (`xcomp`) relations, according to the theory of obligatory control (Farkas, 1988). We introduce a new post-processing option that combines these two patterns, allowing an argument in subject position to be distributed over multiple `xcomp` predicates that are joined by a `conj` relation, as illustrated in Figure 2.8.

2.2.3 Evaluation

We evaluate the original PredPatt (PredPatt v1) and the improved PredPatt (PredPatt v2) on the English Web Treebank (EWT) and the Wall Street Journal

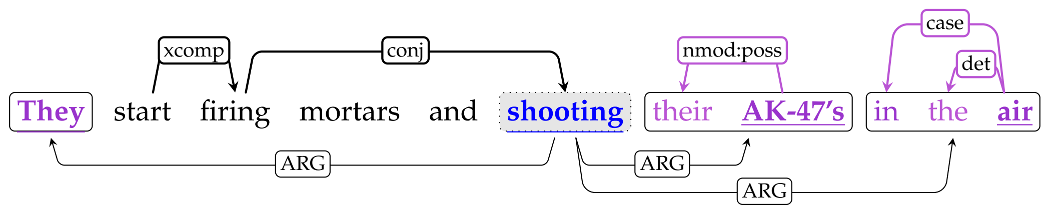


Figure 2.8: Example for borrowing subject from the conjunction of open clausal complement.

corpus (WSJ), and compare their performance with four prominent Open IE systems: OpenIE 4,[6] OLLIE (Mausam et al., 2012), ClausIE (Del Corro and Gemulla, 2013), and Stanford Open IE (Angeli et al., 2015).

Precision-Recall Curve We compare PredPatt with four prominent Open IE systems which are also built for predicate-argument extraction. To allow some flexibility, we compute the precision and recall of different systems by running the scripts[7] used in (Stanovsky and Dagan, 2016), where an automated extraction is matched with a gold extraction based on their token-level overlap. Figure 2.9 and Figure 2.10 show the Precision-Recall Curves for different systems on EWT and WSJ.[8] When tested on EWT which has gold UD parses (Figure 2.9), PredPatt v1 and v2 outperforms the other systems by a significant margin in both precision and recall. When tested on WSJ where only automated UD parses are available (Figure 2.10), ClausIE achieves a recall that is slightly better than PredPatt v1, but PredPatt v2 still shows the best performance across all systems.

[6]OpenIE 4 is available at: `https://github.com/allenai/openie-standalone`.

[7]`https://github.com/gabrielStanovsky/oie-benchmark`.

[8]Studies of PredPatt confidence prediction have been done before, but the evaluated system does not output them. In this evaluation, we assign 1.0 confidence score to all PredPatt extractions.

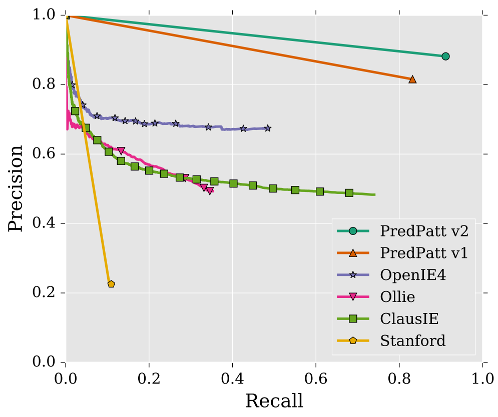


Figure 2.9: Precision-Recall Curve for different systems on EWT w/ gold UD.

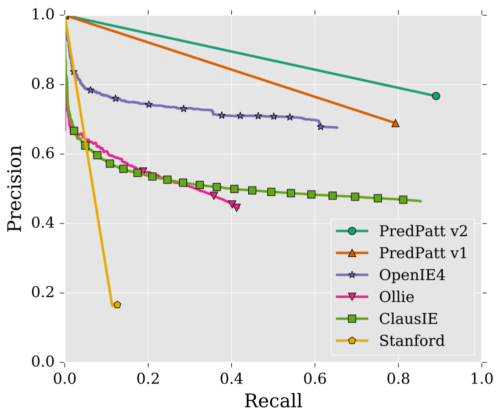


Figure 2.10: Precision-Recall Curve for different systems on WSJ w/ automated UD.

Extraction Head Agreement The rich underlying structure in PredPatt (see Figure 2.6) contains head information for predicates and arguments, which enables a precision-recall metric based on the agreement of head information. Similar to He et al. (2015), we first match an automated predicate with a gold predicate if they both agree on their head.[9] With two matched predicates, we then match an automated argument with a gold argument if the automated argument head is within the gold argument span.

We evaluate the precision and recall by a loose macro measure: For the i-th extractions that have two matched predicates, let the argument set of the gold predicate be A_i, and the argument set of the automated predicate be $\hat{A}_i$. The number of matched arguments is represented by $|A_i \cap \hat{A}_i|$. Then the precision is computed by Precision $= \frac{1}{N}\sum_{i=1}^{N} |A_i \cap \hat{A}_i| / |\hat{A}_i|$, and the recall is computed by Recall $= \frac{1}{N}\sum_{i=1}^{N} |A_i \cap \hat{A}_i| / |A_i|$. Table 2.3 shows the evaluation results of PredPatt v1 and v2 on EWT and WSJ. PredPatt v2 modestly increases the

[9]In the experiment settings, the head of a gold predicate is the verb token in the predicate.

precision by 2.3 on EWT and 0.9 on WSJ, and increases the recall by 1.6 on EWT and 0.2 on WSJ.

	EWT		**WSJ**	
	PredPatt v1	PredPatt v2	PredPatt v1	PredPatt v2
Precision	77.5	79.8 (**+2.3**)	62.1	63.0 (**+0.9**)
Recall	88.0	89.6 (**+1.6**)	84.9	85.1 (**+0.2**)

Table 2.3: Precision and Recall based on the agreement of head information.

Statistics of Argument Span Relations Besides the precision-recall oriented metrics, we impose another metric to further measure the argument span relations. For the *i*-th extractions that have an automated predicate and a gold predicate matched with each other, let an argument in the gold argument set be $\alpha \in A_i$, and an argument in the automated argument set $\beta \in \hat{A}_i$. We categorize the automated extractions into four sets according to their arguments relation to the gold arguments.

$$\begin{aligned}
S_{\text{same}} &= \{(A_i, \hat{A}_i) \mid \forall \alpha \in A_i . \exists \beta \in \hat{A}_i . \text{span}(\alpha) = \text{span}(\beta)\} \\
S_{\text{superset}} &= \{(A_i, \hat{A}_i) \mid \forall \alpha \in A_i . \exists \beta \in \hat{A}_i . \text{span}(\alpha) \subseteq \text{span}(\beta)\} \setminus S_{\text{same}} \\
S_{\text{subset}} &= \{(A_i, \hat{A}_i) \mid \forall \alpha \in A_i . \exists \beta \in \hat{A}_i . \text{span}(\alpha) \supseteq \text{span}(\beta)\} \setminus S_{\text{same}} \\
S_{\text{overlap}} &= \{(A_i, \hat{A}_i) \mid \forall \alpha \in A_i . \exists \beta \in \hat{A}_i . \text{span}(\alpha) \cap \text{span}(\beta) \neq \emptyset\} \setminus \\
&\quad (S_{\text{same}} \cup S_{\text{superset}} \cup S_{\text{subset}})
\end{aligned}$$

Table 2.4 shows the proportion of PredPatt extractions in different sets. As we expected, compared to WSJ, more extractions on EWT fall into S_{same}, which shows that PredPatt works better on gold UD parses. In contrast to PredPatt v1, PredPatt v2 on EWT increases extractions in S_{same} by 12.97%, which contributes to the most increase of S_{subset}; on WSJ, PredPatt v2 decreases extractions in S_{subset} by 13.89%, which leads the major increases of S_{same} and S_{superset}. There are still over 10% extractions not belonging to any of these four sets. Case analysis shows that the inconsistent extractions are mainly caused by incorrect borrowing of arguments for compound predicates or predicates under obligatory control, missing arguments for passive/active verbs that act as adjectival modifiers, etc. These cases are not easily reachable via UD analysis, but leave room for further improvement on PredPatt.

	EWT		**WSJ**	
	PredPatt v1	PredPatt v2	PredPatt v1	PredPatt v2
Same	63.77	76.74 (**+12.97**)	41.56	52.03 (**+10.47**)
Superset	2.74	4.15 (**+1.41**)	8.64	14.10 (**+5.46**)
Subset	18.31	5.82 (**-12.49**)	28.63	14.74 (**-13.89**)
Overlap	0.78	0.39 (**-0.39**)	2.16	1.06 (**-1.10**)
Other	14.40	12.90 (**-1.50**)	19.01	18.07 (**-0.94**)

Table 2.4: Proportion of PredPatt extractions in different sets.

2.3 Summary

In this chapter, we summarize the meaning representation frameworks targeted in this thesis, and review related work for parsing each of these frameworks. Then we introduce PredPatt, a pattern-based predicate-argument extraction framework. We create a large-scale benchmark for predicate-argument extraction by converting manual annotations from PropBank. Based on the benchmark, we improve PredPatt patterns, and compare PredPatt with four prominent Open IE systems. The comparison shows that PredPatt significantly outperforms the other systems. PredPatt is used to provide shallow semantics from raw sentences in Chapter 3 and 4.

Part I

Cross-lingual Semantic Parsing

Chapter 3

MT/IE: Cross-lingual Open Information Extraction

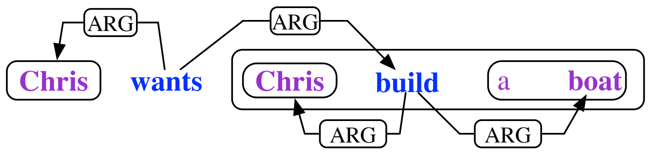
3.1 Introduction

Suppose an English-speaking user is faced with the daunting task of distilling facts from a collection of Chinese documents. One solution is to first translate the Chinese documents into English using a Machine Translation (MT) service, then extract the facts using an English-based Information Extraction (IE) engine. Unfortunately, imperfect translations negatively impact the IE engine, which may have been trained to expect natural English input (Sudo et al., 2004). Another approach is to first run a Chinese-based IE engine and then translate the results, but this relies on IE resources in the source language. Such problems with pipeline systems compound when the IE engine relies on parsers or other analytics as features.

We propose to solve the cross-lingual IE task with a joint approach. Further, we focus on *Open* IE, which allows for an open set of semantic relations between a predicate and its arguments. Open IE in the monolingual setting has

克里斯想 造 一 艘 船 。

(a)



(b)

Figure 3.1: Example of input (a) and output (b) of cross-lingual Open IE.

shown to be useful in a wide range of tasks, such as question answering (Fader et al., 2014), ontology learning (Suchanek, 2014), and summarization (Christensen et al., 2013). A variety of work has achieved compelling results at monolingual Open IE (Banko et al., 2007; Fader et al., 2011; Angeli et al., 2015). But we are not aware of efforts that focus on both the cross-lingual and open aspects of cross-lingual Open IE, despite significant work in related areas, such as cross-lingual IE on a closed, pre-defined set of events/entities (Sudo et al., 2004; Parton et al., 2009; Ji, 2009; Snover et al., 2011; Ji et al., 2016), or bootstrapping of monolingual Open IE systems in multiple languages (Faruqui and Kumar, 2015; Kozhevnikov and Titov, 2013; Plas et al., 2014).

Inspired by the recent success of neural models in machine translation (Kalchbrenner and Blunsom, 2013; Cho et al., 2014; Bahdanau et al., 2014), syntactic parsing (Vinyals et al., 2015a; Choe and Charniak, 2016), and semantic parsing (Dong and Lapata, 2016), we propose variants of sequence-to-sequence models that enable end-to-end cross-lingual Open IE. Essentially, we recast the problem as structured translation: the model encodes natural-language sentences and decodes predicate-argument forms (Figure 3.1). We show that the joint approach outperforms the pipeline on various metrics, and that neural models are critical for the joint approach because of their capability in generating complex open IE patterns.

3.2 Cross-lingual Open IE Framework

Open IE involves the extraction of relations whose schema need not be specified in advance; typically the relation name is represented by the text linking the arguments, which can be identified by manually-written patterns and/or parse trees. We define our extractions based on PredPatt[1] (White et al., 2016), a lightweight tool for identifying predicate-argument structures with a set of Universal Dependencies (UD) based patterns.

PredPatt represents predicates and arguments in a tree structure where a special dependency `ARG` is built between a predicate head token and its arguments' head tokens, and original UD dependencies within predicate phrases and argument phrases are kept. For example, Figure 3.1b shows a tree structure identified by PredPatt from the sentence: *"Chris wants to build a boat.*"

Our framework assumes the availability of a bitext, e.g. a corpus of Chinese sentences and their English translations. We run PredPatt on the target side (e.g. English) to obtain (Chinese sentence, English PredPatt) pairs. This is used to train a cross-lingual Open IE system that maps directly from Chinese sentence to English PredPatt representations. Besides the UD parser required for running PredPatt on the target side, our framework requires no additional resources.

Compared to existing Open IE (Banko et al., 2007; Fader et al., 2011; Angeli

[1]`https://github.com/hltcoe/PredPatt`

et al., 2015), the use of manual patterns on Universal Dependencies means that the rules are interpretable, extensible and language-agnostic, which makes PredPatt a linguistically well-founded component for cross-lingual Open IE. Note that our joint models are agnostic to the IE representation, and can be adapted to other Open IE frameworks.

Next, we propose two methods based on neural sequence-to-sequence variants to tackle cross-lingual Open IE.

3.3 Proposed Method I: Joint Seq2Seq

Our goal is to learn a model which directly maps a sentence input A in the source language into predicate-argument structures output B in the target language. Formally, we regard the input as a sequence $A = x_1, \cdots, x_{|A|}$, and use a *linearized* representation of the predicate-argument structure as the output sequence $B = y_1, \cdots, y_{|B|}$. While tree-based decoders are conceivable (Zhang et al., 2016), linearization of structured outputs to sequences simplifies decoding and has been shown effective in, e.g. Vinyals et al. (2015a), especially when a model with strong memory capabilities (e.g. LSTM's) are employed. Our model maps A into B using a conditional probability which is decomposed as:

$$P(B \mid A) = \prod_{t=1}^{|B|} P(y_t \mid y_1, \cdots, y_{t-1}, A) \tag{3.1}$$

3.3.1 Linearized PredPatt Representations

We begin by defining a linear form for our PredPatt predicate-argument structures. To convert a tree structure such as Figure 3.1b to a linear sequence, we first take an in-order traversal of every node (token). We then label each token with the type it belongs to: p for a predicate token, a for an argument token, p_h for a predicate head token, and a_h for an argument head token. We insert parentheses to either the beginning or the end of an argument, and we insert brackets to either the beginning or the end of a predicate. Figure 3.2 shows the linearized PredPatt for the sentence: *"Chris wants to build a boat.*".

[(Chris:a_h) wants:p_h [(Chris:a_h) build:p_h (a:a boat:a_h)]]

Figure 3.2: Linearized PredPatt Output

To recover the predicate-argument tree structure, we simply build it recursively from the outermost brackets. At each layer of the tree, parentheses help recover argument nodes. The labels a_h and p_h help identify the head token of a predicate and an argument, respectively. We define that an auto-generated linearized PredPatt is malformed if it has unmatched brackets or parentheses, or a predicate (or an argument) has zero or more than one head token.

3.3.2 Seq2Seq Model

Our sequence-to-sequence (Seq2Seq) model consists of an encoder which encodes a sentence input A into a vector representation, and a decoder which learns to decode a sequence of linearized PredPatt output B conditioned on encoded vector.

We adopt a model similar to that which is used in neural machine translation (Bahdanau et al., 2014). The encoder uses an L-layer bidirectional RNN (Schuster and Paliwal, 1997) which consists of a forward RNN reading inputs from x_1 to $x_{|A|}$ and a backward RNN reading inputs in reverse from $x_{|A|}$ to x_1. Let $\overrightarrow{\boldsymbol{h}_i^l} \in \mathbb{R}^n$ denote the forward hidden state at time step i and layer l; it is computed by states at the previous time-step and at a lower layer: $\overrightarrow{\boldsymbol{h}_i^l} = \overrightarrow{f}(\overrightarrow{\boldsymbol{h}_{i-1}^l}, \overrightarrow{\boldsymbol{h}_i^{l-1}})$ where $\overrightarrow{f}$ is a nonlinear LSTM unit (Hochreiter and Schmidhuber, 1997). The lowest layer $\overrightarrow{\boldsymbol{h}_i^0}$ is the word embedding of the token x_i. The backward hidden state $\overleftarrow{\boldsymbol{h}_i^l}$ is computed similarly using another LSTM, and the representation of each token x_i is the concatenation of the top-layers: $\boldsymbol{h}_t = [\overrightarrow{\boldsymbol{h}_i^L}^\intercal, \overleftarrow{\boldsymbol{h}_i^L}^\intercal]^\intercal$.

The decoder is an L-layer RNN which predicts the next token y_i, given all the previous words $\boldsymbol{y}_{<i} = y_1, \cdots, y_{i-1}$ and the context vector $\boldsymbol{c}_i$ that captures the attention to the encoder side (Bahdanau et al., 2014; Luong et al., 2015), computed as a weighted sum of hidden representations: $\boldsymbol{c}_i = \sum_{j=1}^{l} a_{ij}\boldsymbol{h}_j$. The weight a_{ij} is computed by

$$\begin{aligned} a_{ij} &= \frac{\exp(e_{ij})}{\sum_{k=1}^{l} \exp(e_{ik})} \\ e_{ij} &= v_a^\intercal \tanh(\sum_{l=1}^{L} \boldsymbol{W}_a^l s_{i-1}^l + \boldsymbol{U}_a \boldsymbol{h}_j) \end{aligned} \tag{3.2}$$

where $v_a \in \mathbb{R}^n$, $\boldsymbol{W}_a^l \in \mathbb{R}^{n\times n}$ and $\boldsymbol{U}_a \in \mathbb{R}^{n\times 2n}$ are weight matrices.

The conditional probability of the next token y_i is defined as:

$$P(y_i \mid \boldsymbol{y}_{<i}, A) = g(y_i, \boldsymbol{s}_i^L, \boldsymbol{c}_i)$$

$$= \text{softmax}(\boldsymbol{U}_o \boldsymbol{s}_i^L + \boldsymbol{C}_o \boldsymbol{c}_i)[y_i]$$

where $\boldsymbol{U}_o \in \mathbb{R}^{|V_B| \times n}$ and $\boldsymbol{C}_o \in \mathbb{R}^{|V_B| \times 2n}$ are weight matrices.$[j]$ indexes jth element of a vector. s_i^L is the top-layer hidden state at time step i, computed recursively by $\boldsymbol{s}_i^l = f(\boldsymbol{s}_{i-1}^l, \boldsymbol{s}_i^{l-1}, \boldsymbol{c}_i)$ where $\boldsymbol{s}_i^0 = \boldsymbol{W}_B[y_{i-1}]$ is the word vector of the previous token y_{i-1}, with $\boldsymbol{W}_B \in \mathbb{R}^{|V_B| \times n}$ being a parameter matrix.

Training: The objective function is to minimize the negative log likelihood of the target linearized PredPatt given the sentence input:

$$\text{minimize} - \sum_{(A,B) \in \mathcal{D}} \sum_{i}^{|A|} \log P(y_i \mid \boldsymbol{y}_{<i}, A) \tag{3.3}$$

where $\mathcal{D}$ is the batch of training pairs, and $P(y_i \mid \boldsymbol{y}_{<i}, A)$ is computed by Eq.(3.3).

Inference: We use greedy search to decode tokens one by one:

$$\hat{y}_i = \underset{y_i \in V_B}{\text{argmax}}\, P(y_i | \hat{\boldsymbol{y}}_{<i}, A) \tag{3.4}$$

3.4 Proposed Method II: Selective Decoding

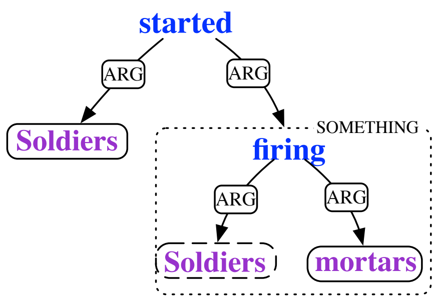
We now show a different formulation of the cross-lingual Open IE task in Figure 3.3. In the above section, we formalize the task as a sequence-to-sequence learning problem by converting the target facts in the tree structure (Figure 3.3c) into a linear form called linearized PredPatt (Figure 3.3d), and

士兵们 开始 发射 迫击炮 。

(a)

Soldiers started firing mortars .

(b)



(c)

[(Soldiers:a_h) started:p_h [(Soldiers:a_h) firing:p_h (mortars:a_h)]]

(d)

[	Soldiers	started	[	Soldiers	firing	mortars	]	]
P	A	P	P	A	P	A	P	P

(e)

Figure 3.3: Example of cross-lingual open IE: Chinese input text (a), English translation (b), English predicate-argument information (c), linearized PredPatt output (d) and output with separated predicate and argument labels (e).

employs a standard encoder-decoder model to address the problem (from Figure 3.3a to Figure 3.3d). In the linearized PredPatt (Figure 3.3d), special labels are appended to tokens as type indications. Brackets and parentheses have to be inserted to delimit predicate and argument spans. Such a workaround could expand the vocabulary space and increase the burdens on the decoder, which is not ideal for sparse data scenarios and limits the overall performance.

To alleviate this issue, in the second method we reformulate cross-lingual open IE as a sequence generation and labeling problem (from Figure 3.3a to Figure 3.3e) by separating the predicate and argument labels from the target linearized PredPatt, and removing unnecessary parentheses. We propose a novel encoder-decoder model which employs a selective decoding mechanism to explicitly model the sequence labeling as well as the sequence generation process. Compared to the first proposed method, this new method substantially reduces the vocabulary space, eases the burden on the decoder, and leads to a significant gain of performance. The natural of the selective decoding mechanism enables a joint training strategy that optimizes sequence generation and labeling simultaneously. In addition, we introduce an adapted beam search algorithm to further improve the prediction quality.

3.4.1 Problem Formulation

Formally we want to directly map a sentence input X in the source language into a sentence Y in the target language, and simultaneously label each token in Y with type information T. For cross-lingual open IE, the types are *predicate* and *argument*. It is important to label types because they are used to annotate predicates and arguments in the generated tokens. We regard the input as a sequence $X = x_1, \cdots, x_{|X|}$, and the output as two sequences (Y, T): (1) the sentence in target language $Y = y_1, \cdots, y_{|Y|}$, and (2) the type information $T = t_1, \cdots, t_{|Y|}$, where $t_i \in \mathcal{T}$ is the label for the token y_i, and $|X|$ and $|Y|$ are the length of the sequence X and Y respectively. Our model maps X into (Y, T) using a conditional probability which is decomposed as:

$$P(Y, T \mid X) = \textstyle\prod_{i=1}^{|Y|} P(y_i, t_i \mid y_{<i}, t_{<i}, X)$$

$$= \prod_{i=1}^{|Y|} P(y_i \mid y_{<i}, t_{\leq i}, X) P(t_i \mid y_{<i}, t_{<i}, X) \tag{3.5}$$

where $y_{<i} = y_1 \cdots y_{i-1}$ and $t_{\leq i} = t_1 \cdots t_i$.

Equation (3.5) can be interpreted as at each decoding time step, the model first decides which type (label) of tokens to generate, and then generates a token for that type.

3.4.2 Selective Decoding Model

To learn the factored conditional probabilities as shown in Equation (3.5), we propose a novel encoder-decoder model with the selective decoding mechanism: on the encoder side, an input sentence X is encoded into vector representations; on the decoder side, the selective decoding mechanism employs multiple decoders each of which learns the conditional probability of decoding a specific type of token (i.e., $P(y_i \mid y_{<i}, t_{\leq i}, X)$), and a selector learning to decide which type of decoder to use (i.e., $P(t_i \mid y_{<i}, t_{<i}, X)$) at each decoding time step.

Figure 3.4 illustrates the selective decoding process for the example shown in Figure 3.3e. s_0 is the initial decoder hidden state initialized by the last hidden state of the encoder. Special tokens $\langle bos \rangle$ and $\langle eos \rangle$ are added to the beginning and the end of the sequence to indicate the start and the finish of decoding. The connections at each decoding time step are dynamically

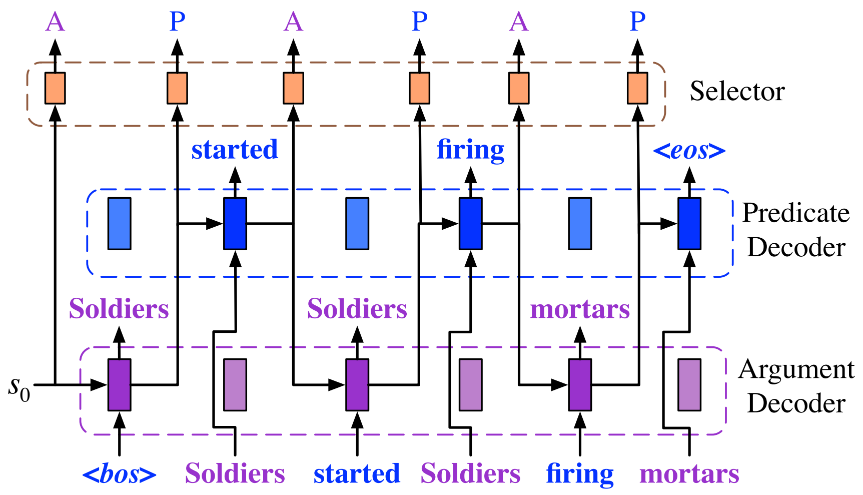


Figure 3.4: Selective decoding process.
(Brackets and attention layers are omitted.)

changing according to the decision of the selector. Specifically, at the decoding time step i, firstly the selector on the top decides to use which type of decoder $D_{t_i} \in \{D_{\mathsf{P}}, D_{\mathsf{A}}\}$[2], and then the decoder D_{t_i} decodes the token y_i which is naturally given the label t_i.

In addition to distinguish between labels, the multiple decoders used by the selective decoding mechanism has two prominent advantages over the single standard RNN decoder: (1) Multiple decoders learn different conditional probability distributions for predicate and argument generation respectively. For instance, given the same input token "*wanted*", the predicate decoder would like to next generate tokens such as "*to*" and "*by*" which starts a prepositional phrase, whereas the argument decoder would be in favor of tokens such as "*a*" and "*him*" which starts a direct object. (2) Multiple decoders reduce the decoder vocabulary size, which eases the burden of sequence generation. Moreover, we propose an efficient architecture that supports batch training of

[2] D_{P} stands for the predicate decoder, and D_{A} for the argument decoder.

the model. The details of the architecture are described in the **Decoder with Selective Decoding** section.

Encoder The encoder employs a bi-directional RNN (Schuster and Paliwal, 1997) to encode the input sequence $X = x_1, \cdots, x_{|X|}$ into a sequence of hidden states $\boldsymbol{h} = h_1, \cdots, h_{|X|}$. Each hidden state h_i in $\boldsymbol{h}$ is a concatenation of a left-to-right hidden state $\overrightarrow{h_i} \in \mathbb{R}^n$ and a right-to-left hidden state $\overleftarrow{h_i} \in \mathbb{R}^n$,

$$h_i = \begin{bmatrix} \overleftarrow{h}_i \\ \\ \overrightarrow{h}_i \end{bmatrix} = \begin{bmatrix} \overleftarrow{f}(x_i, \overleftarrow{h}_{i+1}) \\ \\ \overrightarrow{f}(x_i, \overrightarrow{h}_{i-1}) \end{bmatrix},$$

where $\overleftarrow{f}$ and $\overrightarrow{f}$ are two L-layer stacked LSTMs units (Hochreiter and Schmidhuber, 1997).

Decoder with Selective Decoding Unlike the single standard RNN decoder, which recurrently uses the same decoder to generate tokens, our model dynamically selects different decoders at each decoding time step to generate tokens (Figure 3.4). However, since the decoding path may be different for each input sequence X, directly running the selective decoding process suffers from a key technical issue: it does not support batched computation, making them slow and unwieldy for large-scale NLP tasks (Bowman et al., 2016).

To address this issue, we introduce a general decoding architecture that is applicable to all selective decoding processes. The detailed connection in the architecture is shown in Figure 3.5. At each decoding time step, the model feeds the input token and the previous hidden state to all types of decoders, and use a mask vector created by the selector to select the decoder output to generate tokens and update the hidden state.

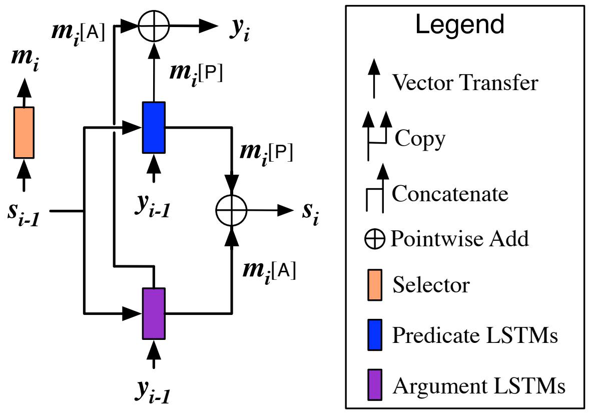


Figure 3.5: Detailed connection at a decoding step. (Attention layers are ommited.)

Formally, let $s_i \in \mathbb{R}^n$ denote the hidden state at decoding time step i. The last left-to-right hidden state $\overrightarrow{h}_{|X|}$ from the encoder is used to initialize the first hidden state s_0 in the decoder.

At the decoding time step i, given the sequence of encoder hidden states $\boldsymbol{h}$ and the previous decoder hidden state s_{i-1}, the selector computes the conditional probability of t_i (i.e., the type of decoder to use) as:

$$P(t_i \mid y_{<i}, t_{<i}, X) = g(t_i, s_{i-1}, \boldsymbol{h}) = \text{softmax}(U_o s_{i-1} + C_o c_{i-1} + b_o)[t_i], \quad (3.6)$$

where $U_o \in \mathbb{R}^{|\mathcal{T}| \times n}$, $C_o \in \mathbb{R}^{|\mathcal{T}| \times n}$ and $b_o \in \mathbb{R}^{|\mathcal{T}|}$ are weight matrices and bias.[3] $[t_i]$ indexes the element of a vector that corresponds to the type t_i.

The context vector c_{i-1} captures the attention to the encoder side (Bahdanau et al., 2014; Luong et al., 2015), computed as a weighted sum of encoder

[3] $|\mathcal{T}|$ is the number of token types. In the example shown in Figure 3.5, $\mathcal{T} = \{\mathsf{P}, \mathsf{A}\}$, and $|\mathcal{T}| = 2$.

hidden states: $c_{i-1} = \sum_j^{|X|} a_{(i-1)j} h_j$. The weight $a_{(i-1)j}$ is computed by:

$$a_{(i-1)j} = \frac{\exp\left(\text{score}(s_{i-1}, h_j)\right)}{\sum_{j'=1}^{|X|} \exp\left(\text{score}(s_{i-1}, h_{j'})\right)}, \tag{3.7}$$

where $\text{score}(s_{i-1}, h_j) = s_{i-1}^{\intercal} W_a h_j$, and $W_a \in \mathbb{R}^{n \times 2n}$ is a transform matrix.

According to the selector output $P(t_i \mid y_{<i}, t_{<i}, X)$, a mask vector $\boldsymbol{m}_i$ is created, which is used to mask out the decoders' hidden states,

$$\boldsymbol{m}_i[t_i] = \begin{cases} 1, \text{if } t_i = \underset{t'_i \in \mathcal{T}}{\operatorname{argmax}} P(t'_i \mid y_{<i}, t_{<i}, X) \\ \\ 0, \text{otherwise} \end{cases}$$

Then the hidden state s_i for the decoding time step i is computed by:

$$s_i = \sum_{t_i \in \mathcal{T}} \boldsymbol{m}_i[t_i] f_{t_i}(y_{i-1}, s_{i-1}, c_i)$$

where c_i is the context vector capturing the *attention*, computed in the same way as Equation (3.7). f_{t_i} is L-layer stacked LSTMs for the type t_i. In Figure 3.5, there is an L-layer stacked LSTMs for generating predicate tokens f_{P}, and another L-layer stacked LSTMs for generating argument tokens f_{A}. They have untied parameters.

The conditional probability of the token y_i with the type t_i is defined as:

$$P(y_i \mid y_{<i}, t_{\leq i}, X) = g'(y_{i-1}, s_{i-1}, \boldsymbol{h}, \boldsymbol{m}_i) = \text{softmax}(U'_o s_i + C'_o c_i + b'_o)[y_i], \tag{3.8}$$

where $U'_o \in \mathbb{R}^{|\mathcal{V}| \times n}$, $C'_o \in \mathbb{R}^{|\mathcal{V}| \times n}$ and $b_o \in \mathbb{R}^{|\mathcal{V}|}$ are weight matrices and bias.[4]

Training In the training procedure, our optimization objective is to minimize

[4] $|\mathcal{V}|$ is the vocabulary size of the target language.

the negative log-likelihood of the sequence Y and its type information T given the input sequence X over the training data, defined as:

$$\text{minimize} - \sum_{(X,Y,T)\in\mathcal{D}} \log P(Y,T \mid X)$$

According to Equation (3.5), the log-likelihood $\log P(Y,T \mid X)$ can be decomposed as:

$$\sum_{i=1}^{|Y|} [\log P(y_i \mid y_{<i}, t_{\leq i}, X) + \log P(t_i \mid y_{<i}, t_{<i}, X)],$$

where $P(y_i \mid y_{<i}, t_{\leq i}, X)$ models the sequence generation process, and $P(t_i \mid y_{<i}, t_{<i}, X)$ models the sequence labeling process. They are computed by Equations (3.8) and (3.6) respectively. The decomposition of the log-likelihood into these two parts enables a joint optimization for the sequence generation and labeling process simultaneously.

We use the Adam optimizer (Kingma and Ba, 2014) and mini-batch gradient to solve this optimization problem. To prevent overfitting, we apply dropout operators (Srivastava et al., 2014) to non-recurrent connections between LSTM layers.

Inference In the inference procedure, we predict the sequence Y and its type information T for an input sequence X according to:

$$(\hat{Y}, \hat{T}) = \operatorname*{argmax}_{(Y',T')\in\mathcal{V}^{|Y'|}\times\mathcal{T}^{|Y'|}} P(Y', T' \mid X)$$

$\mathcal{V}^{|Y'|} \times \mathcal{T}^{|Y'|}$ is the set of all possible (Y', T') pairs. And $(\hat{Y}, \hat{T})$ can be directly converted to the form of linearized PredPatt which is used for evaluation.

However, it is impractical to iterate over all these (Y', T') pairs during inference. Instead, we use beam search to generate tokens and labels. The beam is used to increase the search space for the sequence Y in the target language. At each decoding time step, we first greedily select the type of decoder, and then generate candidate tokens from the selected decoder to update the beam. When the special token $\langle eos \rangle$ is generated, we remove the candidate sequence from the beam.

3.5 *Halo*: Learning Semantics-Aware Representations

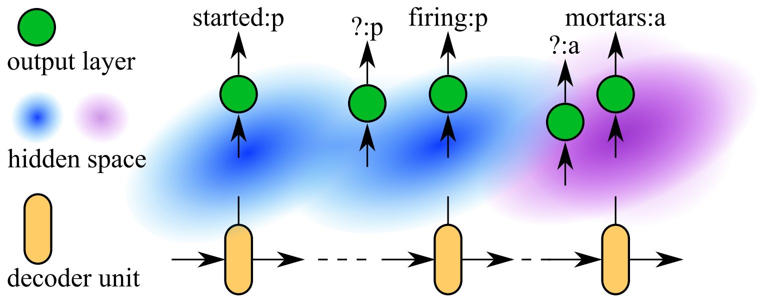


Figure 3.6: Visualization of *Halo* method.

In terms of training, beside maximum likelihood estimation (MLE), we propose *Halo* – a semantics-aware representation learning method for cross-lingual Open IE. As each member in the target vocabulary is essentially either predicate or argument, a random perturbation on the hidden state should still be able to yield a token with the same semantic structure tag. This inductive bias motivates an extra term in training objective, as shown in Figure 3.6, which enforces the surroundings of any learned hidden state to generate tokens with the same semantic structure tag (either predicate or argument) as the centroid. We call this technique *Halo*, because the process of each hidden state taking up its surroundings is analogous to how the halo is formed around the sun. The method is believed to help the model generalize better, by learning more semantics-aware and noise-insensitive hidden states without introducing extra parameters.

Proposed Learning method Our method adopts a property of this task—the vocabulary $\mathcal{V}$ is partitioned into $\mathcal{P}$, set of predicates that end with ":p", and $\mathcal{A}$, set of arguments that end with ":a". As a neural model would summarize everything known up to step t into hidden state $\mathbf{h}_t$, would a perturbation $\mathbf{h}'_t$ around $\mathbf{h}_t$ still generate the same token y_t? This bias seems too strong, but we can still reasonably assume that $\mathbf{h}'_t$ would generate a token with the same semantic structure tag (i.e. a predicate or argument). That is, the prediction made by $\mathbf{h}'_t$ should end with ":p" if y_t is a predicate, and with ":a" otherwise.

This inductive bias provides us with another level of supervision. Suppose that at step t, a neighboring $\mathbf{h}'_t$ is randomly sampled around $\mathbf{h}_t$, and is then used to generate a distribution $\mathbf{p}'_t$. Then we can get a distribution $\mathbf{q}'_t$ over $\mathcal{C} = \{\text{predicate}, \text{argument}\}$, by summing all the probabilities of predicates and those of arguments:

$$q'_{t,\text{predicate}} = \sum_{v \in \mathcal{P}} p'_{t,v} \tag{3.9a}$$

$$q'_{t,\text{argument}} = \sum_{v \in \mathcal{A}} p'_{t,v} \tag{3.9b}$$

This aggregation is shown in Figure 3.7. Then the extra objective is $\ell' = \sum_{t=1}^{T} \log q'_{t,c_t}$, where $c_t = \text{predicate}$ if the target token $y_t \in \mathcal{P}$ (i.e. ending with ":p") and $c_t = \text{argument}$ otherwise.

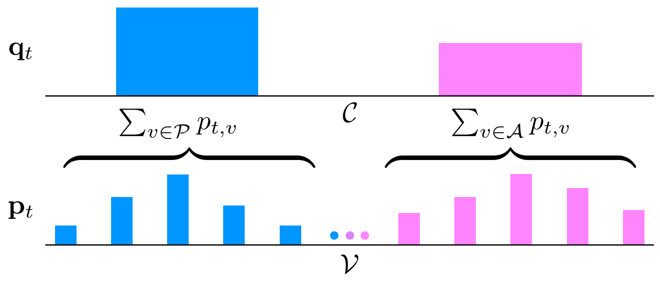

$\mathbf{q}_t$
$\mathcal{C}$
$\sum_{v\in\mathcal{P}} p_{t,v}$
$\sum_{v\in\mathcal{A}} p_{t,v}$
$\mathbf{p}_t$
$\mathcal{V}$

Figure 3.7: Visualization of how $\mathbf{q}$ (distribution over $\mathcal{C}$) is obtained by aggregating $\mathbf{p}$ (distribution over $\mathcal{V}$).

Therefore, we get the joint objective to maximize by adding ℓ and ℓ':

$$\ell + \ell' = \sum_{t=1}^{T} \log p_{t,y_t} + \sum_{t=1}^{T} \log q'_{t,c_t} \tag{3.10}$$

which enables the model to learn more semantics-aware and noise-insensitive hidden states by enforcing the hidden states within a region to share the same semantic structure tag.[5]

Sampling Neighbors Sampling a neighbor around $\mathbf{h}_t$ is essentially equivalent to adding noise to it. Note that in a LSTM decoder that previous work used, $\mathbf{h}_t \in (-1,1)^D$ because $\mathbf{h}_t = \mathbf{o}_t \odot \tanh(\mathbf{c}_t)$ where $\mathbf{o}_t \in (0,1)^D$ and $\tanh(\mathbf{c}_t) \in (-1,1)^D$. Therefore, extra work is needed to ensure $\mathbf{h}'_t \in (-1,1)^D$. For this purpose, we follow the recipe[6]:

- Sample $\mathbf{h}''_t \in (-1,1)^D$ by independently sampling each entry from an uniform distribution over $(-1,1)$;
- Sample a scalar $\lambda_t \in (0,1)$ from a Beta distribution $B(\alpha, \beta)$ where α and

[5]One can also sample multiple, rather than one, neighbors for one hidden state and then average their $\log q'_{t,c_t}$. In our experimental study, we only try one for computational cost and found it effective enough.

[6]Alternatives do exist. For example, one can transform $\mathbf{h}_t$ from $(-1,1)^D$ to $(-\infty,\infty)^D$, add random (e.g. Gaussian) noise in the latter space and then transform back to $(-1,1)^D$. These tricks are valid as long as they find neighbors within the same space $(-1,1)^D$ as $\mathbf{h}_t$ is.

β are hyperparameters to be tuned;

- Compute $\mathbf{h}'_t = \mathbf{h}_t + \lambda_t(\mathbf{h}''_t - \mathbf{h}_t)$ such that $\mathbf{h}'_t \in (-1, 1)^D$ lies on the line segment between $\mathbf{h}_t$ and $\mathbf{h}''_t$.

Note that the sampled hidden state $\mathbf{h}'_t$ is only used to compute $\mathbf{q}'_t$, but not to update the LSTM hidden state, i.e., $\mathbf{h}_{t+1}$ is independent of $\mathbf{h}'_t$.

Roles of Hyperparameters The *Halo* technique adds an inductive bias into the model, and its magnitude is controlled by λ_t:

- $\lambda_t \in (0, 1)$ to ensure $\mathbf{h}'_t \in (-1, 1)^D$;
- $\lambda_t \to 0$ makes $\mathbf{h}'_t \to \mathbf{h}_t$, thus providing no extra supervision on the model;
- $\lambda_t \to 1$ makes $\mathbf{h}'_t$ uniformly sampled in entire $(-1, 1)^D$, and causes underfitting just like a L-2 regularization coefficient goes to infinity.

We sample a valid λ_t from a Beta distribution with $\alpha > 0$ and $\beta > 0$, and their magnitude can be tuned on the development set:

- When $\alpha \to 0$ and β is finite, or α is finite and $\beta \to \infty$, we have $\lambda_t \to 0$;
- When $\alpha \to \infty$ and β is finite, or α is finite and $\beta \to 0$, we have $\lambda_t \to 1$;
- Larger α and β yield larger variance of λ_t, and setting λ_t to be a constant is a special case that $\alpha \to \infty$, $\beta \to \infty$ and α/β is fixed.

Besides α and β, the way of partitioning $\mathcal{V}$ also serves as a knob for tuning the bias strength. Although on this task, the predicate and argument tags naturally partition the vocabulary, we are still able to explore other possibilities. For example, an extreme is to partition $\mathcal{V}$ into $|\mathcal{V}|$ different singletons, meaning that $\mathcal{C} = \mathcal{V}$—a perturbation around $\mathbf{h}_t$ should still predict the same token. But this extreme case does not work well in our experiments, verifying the importance of the semantic structure tags on this task.

3.6 Related Work

The models we propose in this paper are adapted from the RNN encoder-decoder architectures which have been successfully applied to a wide range of NLP tasks such as machine translation (Kalchbrenner and Blunsom, 2013; Cho et al., 2014; Bahdanau et al., 2014), image description generation (Karpathy and Fei-Fei, 2015; Vinyals et al., 2015b), syntactic parsing (Vinyals et al., 2015a), question answering (Hermann et al., 2015), summarization (Rush et al., 2015), and semantic parsing (Dong and Lapata, 2016).

The selective decoding mechanism can be viewed as having different types of decoders stacking together and adding a hard gate to the RNN unit, through which the bit of information will be either totally kept or dropped. It may seem redundant since the RNN gated unit already has the sophisticated gating mechanism such as the GRU unit (Cho et al., 2014) and the LSTM unit (Hochreiter and Schmidhuber, 1997). However, we think that the selective decoding mechanism is a complement to the gated unit: rather than having a soft pointwise control, the selective decoding mechanism adopts a hard vectorwise control to explicitly select a certain type of information which corresponds to the predicate or the argument by keeping one and dropping the others, whereas the GRU/LSTM gated unit itself learns to memorize long short-term dependencies. Similar mechanisms have been used in neural machine translating (Tu et al., 2016) and image caption generation (Xu et al., 2015) to explicitly control the influence from source or target contexts. The experiments in § 3.7 also confirms our point: our model using the selective decoding mechanism significantly improves the performance, compared to the standard encoder-decoder model.

Regarding to open IE systems for generating training data, PredPatt has shown promising performance on large-scale open IE benchmarks (Zhang et al., 2017e). Compared to other open IE systems (Banko et al., 2007; Fader et al., 2011; Angeli et al., 2015), PredPatt uses manual language-agnostic patterns on UD, which makes it a well-founded component across languages. Additionally, the underlying structure constructed by PredPatt has been shown to be a well-formed syntax-semantics interface (Zhang et al., 2017d).

Our proposed learning method *Halo* can be understood as a data augmentation technique (Chapelle et al., 2001; Van der Maaten et al., 2013; Srivastava et al., 2014; Szegedy et al., 2016; Gal and Ghahramani, 2016). Such tricks have been used in training neural networks to achieve better generalization, in applications like image classification (Simard et al., 2000; Simonyan and Zisserman, 2015; Arpit et al., 2017; Zhang et al., 2017a) and speech recognition (Graves et al., 2013; Amodei et al., 2016). *Halo* differs from these methods because 1) it makes use of the task-specific information—vocabulary is partitioned by semantic structure tags; and 2) it makes use of the human belief that the hidden representations of tokens with the same semantic structure tag should stay close to each other.

3.7 Experiments

We describe the data for evaluation, hyperparameters, comparing approaches and evaluation results.[7]

3.7.1 Hyperparameters

Our proposed models are trained using the Adam optimiser (Kingma and Ba, 2014), with mini-batch size 64 and step size 200. Both encoder and decoder have 2 layers and hidden state size 512, but different LSTM parameters sampled from $\mathcal{U}$(-0.05,0.05). Vocabulary size is 40K for both sides. Dropout (rate=0.5) is applied to non-recurrent connections (Srivastava et al., 2014). Gradients are clipped when their norm is bigger than 5 (Pascanu et al., 2013). We use sampled softmax to speed up training (Jean et al., 2015). The number of epochs is 20. Early stopping is used to avoid overfitting. The beam size is 5. Before feeding into the encoder, we reverse the input sentences (Sutskever et al., 2014).

In experiments of *Halo*, instead of using the full vocabularies, we set a minimum count threshold for each dataset, to replace the rare words by a special out-of-vocabulary symbol. These thresholds were tuned on dev sets. The Beta distribution is very flexible. In general, its variance is a decreasing function of $\alpha + \beta$, and when $\alpha + \beta$ is fixed, the mean is an increasing function of α. In our experiments, we fixed $\alpha + \beta = 20$ and only lightly tuned α on dev sets. Optimal values of α stay close to 1.

[7]The code is available at `https://github.com/sheng-z/cross-lingual-open-ie`.

3.7.2 Chinese-English Open IE

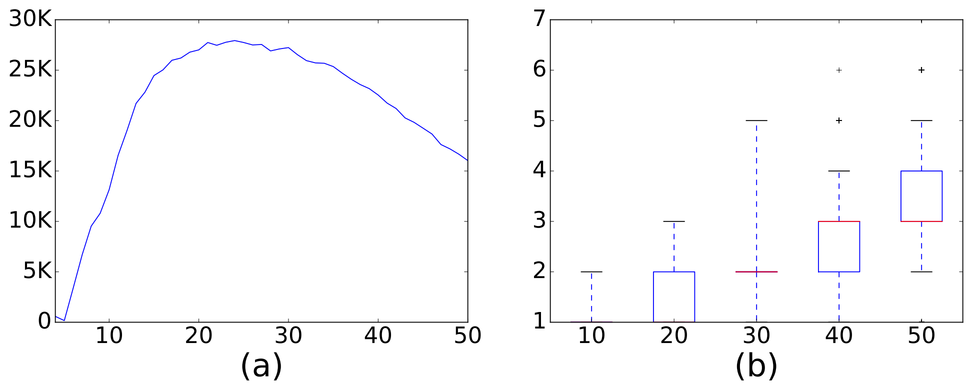


Figure 3.8: Data Statistics: (a) Number of data pairs with respect to the lengths of English linearized PredPatt; (b) Boxplot of numbers of English predicate with respect to the lengths of English linearized PredPatt.

Data We choose Chinese as the source language and English as the target language. To prepare the data for evaluation, we first collect about 2M Chinese-English parallel sentences[8]. We then tokenize Chinese sentences using Stanford Word Segmenter (Chang et al., 2008), and generate English linearized PredPatt by running SyntaxNet Parser (Andor et al., 2016) and PredPatt (White et al., 2016) on English sentences. After removing long sequences (length>50), we result in 990K pairs of Chinese sentences and English linearized PredPatt, which are then randomly divided for training (950K), validation (10K) and test (40K). Figure 3.8 shows the statistics of the data. Note that in general, the linearized PredPatt sequences are not short, and can contain multiple predicates.

Comparisons We compare the following methods: (1) **Pipeline-S** consists of a Moses system (Koehn et al., 2007) that translates Chinese sentence to English

[8]The data comes from the GALE project (Cohen, 2007); the largest bitexts are LDC2007E103 and LDC2006G05

sentence, followed by SyntaxNet Parser (Andor et al., 2016) for Universal Dependency parsing on English, and PredPatt (White et al., 2016) for predicate-argument identification; (2) **Pipeline-N** is the same as Pipeline-S except that the Moses system is replaced by OpenNMT (Klein et al., 2017), a neural machine translation system. (3) **Joint Moses** trains a phrase-based machine translation system Moses directly on on the Chinese-English Open IE dataset; (4) **Joint Seq2Seq** trains a standard encoder-decoder model on the same dataset; (5) ***Halo*** trains a standard encoder-decoder model using *Halo*; and (6) **Selective Decoding** employs the proposed selective decoding mechanism in the encoder-decoder architecture.

Evaluation Metrics We regard the generation of linearized PredPatt or linearized predicates[9] as a translation problem, and use BLEU score (Papineni et al., 2002) for evaluation. We also evaluate predicates and arguments in the same vein as event detection evaluation using the token-level F_1 score (Liu et al., 2015). Token-level F_1 gives partial credits to partial matches.

Evaluation using BLEU Table 3.1 shows the cased BLEU scores of linearized PredPatt and linearized predicates on the test set. Except Joint Moses, all other joint approaches based on neural Seq2Seq variants surpass pipeline approaches. Joint Seq2Seq outperforms the best pipeline (Pipeline-N) by 0.91 BLEU for linearized PredPatt and 2.96 BLEU for linearized predicates. *Halo* and Selective Decoding further improve the performance by a significant margin. The best performance is from Selective Decoding: 23.88 BLEU for

[9]In linearized predicates, arguments are replaced by placeholders. For example, the linearized PredPatt in Figure 3.3d becomes "[?arg wants:p_h Sth:= [?arg build:p_h ?arg]]" after replacement.

linearized PredPatt and 25.42 BLEU for linearized predicates.

Approach	Linearized PredPatt	Linearized Predicate
Pipeline-S	17.19	17.24
Pipeline-N	18.03	18.59
Joint Moses	18.34	16.43
Joint Seq2Seq	18.94	21.55
Halo	23.18	-
Selective Decoding	**23.88**	**25.42**

Table 3.1: Evaluation results (BLEU) of linearized PredPatt and linearized predicates on the test set.

Selective Decoding explicitly models sequence generation and sequence labeling, which enables a standalone evaluation of the sequence generation process (i.e., the final output without predicate and argument labels). To make a baseline comparison, we train an OpenNMT system (Klein et al., 2017) directly on the same data ignoring the labels.

OpenNMT	Selective Decoding
24.92	**25.16**

Table 3.2: Evaluation results (BLEU) of sequence generation on the test set.

Table 3.2 shows the BLEU score of sequence generation on the test set. Selective Decoding achieves higher BLEU than OpenNMT. It demonstrates that the selective decoding mechanism learning with extra labels helps improve the quality of sequence generation. We also notice that the BLEU score (25.16 in Table 3.1) of the final linearized PredPatt from Selective Decoding is even higher than OpenNMT (24.92 in Table 3.2). Hence, we can draw a conclusion that simply placing a labeler atop the OpenNMT system to tackle the cross-lingual open IE problem will not narrow the gap in BLEU between itself and our Selective Decoding approach.

Approach	Predicate	Argument
Pipeline-S	24.24	33.54
Pipeline-N	24.41	33.51
Joint Moses	25.11	38.90
Joint Seq2Seq	25.79	34.44
Halo	30.85	**41.23**
Selective Decoding	**31.71**	40.81

Table 3.3: Evaluation results (F_1) of predicates and arguments on the test set.

Evaluation using F_1 We compute the token-level F_1 score (Liu et al., 2015) of predicates and arguments. As shown in Table 3.3, *Halo* and Selective Decoding substantially improves the F_1 score of both predicates and arguments over the baselines. They both outperform the baselines by over 5% F_1 for predicates and 6% F_1 for arguments.

Pipeline-S	Pipeline-N	Joint Moses	Joint Seq2Seq	Sel. Decoding
5,965	6,014	33,178	557	**53**

Table 3.4: Number of unrecoverable outputs.

Recoverability We compute the number of the linearized PredPatt outputs from which the tree structure representation can not be recovered, including the empty outputs and the outputs which have unmatched brackets, or have zero or multiple heads for an argument or a predicate. Table 3.4 shows the results: Around 15% of pipeline outputs are empty. Joint Moses generates no empty output, but a large amount (84%) of its outputs is malformed. In contrast, Joint Seq2Seq generates very few malformed ones (1%). Selective Decoding further reduces the number of unrecoverable outputs by one order of magnitude (0.1%).

Analysis We analyze the difference between Selective Decoding and the previous best approach Joint Seq2Seq through a plot of BLEU scores for the linearized PredPatt on the test set with respect to the lengths of the reference. As shown in Figure 3.9, when the reference length is greater than 20, the linearized PredPatt generated by Selective Decoding gets notably better BLEU scores, especially for the reference length around 30. However, when the reference length is shorter than 11, the performance of Selective Decoding drops below the Joint Seq2Seq approach.

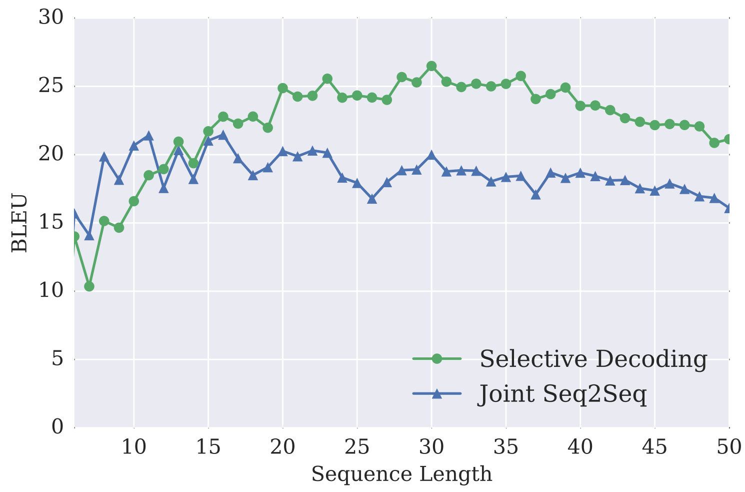


Figure 3.9: BLEU scores of the linearized PredPatt on the test set w.r.t. the lengths of the references.

To explain this performance drop, we randomly sample an example from the test set, where the reference length is shorter than 11, and the BLEU score of the linearized PredPatt generated by Joint Seq2Seq is higher than Selective

Decoding. The example is shown in Table 3.5.

Input sentence and its English translation:

> 我 哪怕 有 千分之一 的 希望 呢 , 我 死活 都 要 给 他 做 最 后 的
> (*Even if there was only a one thousandth of a hope , er , live or die I would give him my all.*)

Reference[10]:

(1) (**I**) would **give** (**him**) (my **all**)

Selective Decoding:

(1) Even if (**we**) **have** (a _UNK per cent **hope**)

(2) uh , (**I**) would **have** SOMETHING[11]

(3) SOMETHING := (**I**) **give** (**him**) (the final **thing**)

Joint Seq2Seq:

(1) (**I**) **wish** (**everyone**) (last **hope**)

Table 3.5: Example outputs with the reference length shorter than 11.

In this example (Table 3.5), the Chinese input sentence has a grammatical error: the object modified by "最后 的" is missing. Additionally, the reference linearized PredPatt output in this example is incomplete: the fact related to the concessive clause is missing. Although here Joint Seq2Seq gets the better BLEU score against the incomplete reference, Selective Decoding is able to better generlize over the train data: the facts it generates are much closer to the original input sentence, and even better than the reference.

Another example where the reference length is greater than 20 is shown in

[10]The predicate tokens are colored blue, and the argument tokens are colored purple. Head tokens are underlined in bold. Token labels and brackets are omitted.

[11]"SOMETHING" is a special argument used to indicate that the argument is a proposition.

Input sentence and its English translation:

结果 , 民主党 失去 了 列举 布什 " 罪状 " 的 良机 ,

(*As a result, the Democratic party lost a good opportunity to list the ' charges ' against Bush.*

Reference:

(1) As (a **result**) , (the Democratic **party**) **lost** (a good **opportunity**)

(2) (a good **opportunity**) **list** (the ' **charges** ' against Bush)

Selective Decoding:

(1) As (a **result**) , (the Democratic **Party**) **lost** (the good **opportunity**)

(2) (the good **opportunity**) **cite** (**Bush**)

Joint Seq2Seq:

(1) (The **result**) **is** SOMETHING

(2) SOMETHING := (the Democratic **Party**) **lost** (his **opportunity**)

(3) (his **opportunity**) **give** (**him**) (good **opportunity**)

Table 3.6: Example outputs with the reference length longer than 20.

Table 3.6. In this example, Selective Decoding generates the same number of facts as the reference, and the meaning of the facts is closer to the reference than Joint Seq2Seq: though not perfect, Selective Decoding captures "列举 布什 ' 罪状 '" ("list the 'charges' against Bush") by generating "**cite** (**Bush**)", whereas Joint Seq2Seq fails to generate any thing related.

3.7.3 Low-resource Scenarios

One of the goals of cross-lingual open IE is to extract facts from languages for which few NLP resources and tools are available, and represent the facts in the language for which plenties of resources and tools can be used. Therefore, we extend the experiments to cross-lingual open IE from 5 languages to English in a low-resource setting.

Task	#Train	#Valid	#Test
uzb-eng	31,581	1,373	1,373
tur-eng	20,774	903	903
amh-eng	12,140	527	527
som-eng	10,702	465	465
yor-eng	5,787	251	251

Table 3.7: Number of data used for cross-lingual open IE in low-resource scenarios.

Datasets To prepare the experiment datasets, we first collect bitexts from DARPA LORELEI language packs (Strassel and Tracey, 2016). The source languages of the bitexts are Uzbek, Turkish, Amharic, Somali, and Yoruba.[12]

We then run a process similar to Chinese-English cross-lingual Open IE to generate pairs of source-language sentences and English linearized PredPatt: first, we employ SyntaxNet Parser (Andor et al., 2016) to generate Universal Dependency parses for the English sentences, and then run PredPatt (White et al., 2016) to generate English linearized PredPatt from the Universal Dependency parses. We remove empty sequences and very long sequences (length>50) in the pairs, and randomly split them into training, validation and test sets in the ratio of 23:1:1. The detailed number of pairs for each experiment is shown in Table 3.7.

Evaluation Results Table 3.8 shows the evaluation results using BLEU. Both

[12]These bitexts are from LDC2016E29 (uzb-eng); LDC2014E115 (tur-eng); LDC2016E86 and LDC2016E87 (amh-eng); LDC2016E90 and LDC2016E91 (som-eng); LDC2016E104 and LDC2016E105 (yor-eng).

Halo and Selective Decoding outperforms the Joint Seq2Seq approach, demonstrating that they are better at overcoming data sparsity. *Halo* experiment results are not reported on two low resource datasets Amharic and Yoruba, because $\alpha = 0$ in Halo was found optimal on the dev sets. In such cases, this regularization was not helpful so no comparison need be made on the held-out test sets.

Task	Joint Seq2Seq	*Halo*	Selective Decoding
uzb-eng	8.66	**12.95**	10.76
tur-eng	7.18	**10.21**	7.47
amh-eng	7.18	-	**8.37**
som-eng	10.61	**14.62**	13.06
yor-eng	11.31	-	**12.19**

Table 3.8: Evaluation results in low-resource cross-lingual open IE scenarios: BLEU of linearized PredPatt.

3.8 Conclusions

We focus on the problem of cross-lingual open IE, and propose joint solutions based on neural sequence-to-sequence models. Our joint approaches outperforms the traditional pipeline solutions by a large margin. We present a simple and effective training technique *Halo*, enabling the learned hidden states to be more aware of semantics and robust to random noise. Regarding to future work, we are interested in cross-lingual semantic parsing whose target representations contains rich information about predicates and arguments, and can be used to facilitate semantic analysis in a cross-lingual setting.

Chapter 4

Cross-lingual Decompositional Semantic Parsing

4.1 Introduction

We are concerned here with representing the semantics of multiple natural languages in a single semantic analysis. Renewed interest in semantic analysis has led to a surge of proposed new frameworks, e.g., GMB (Basile et al., 2012), AMR (Banarescu et al., 2013), UCCA (Abend and Rappoport, 2013), and UDS (White et al., 2016), as well as further calls to attend to existing efforts, e.g., Episodic Logic (Schubert and Hwang, 2000; Schubert, 2000; Hwang and Schubert, 1994; Schubert, 2014), or Discourse Representation Theory (Kamp, 1981; Heim, 1988).

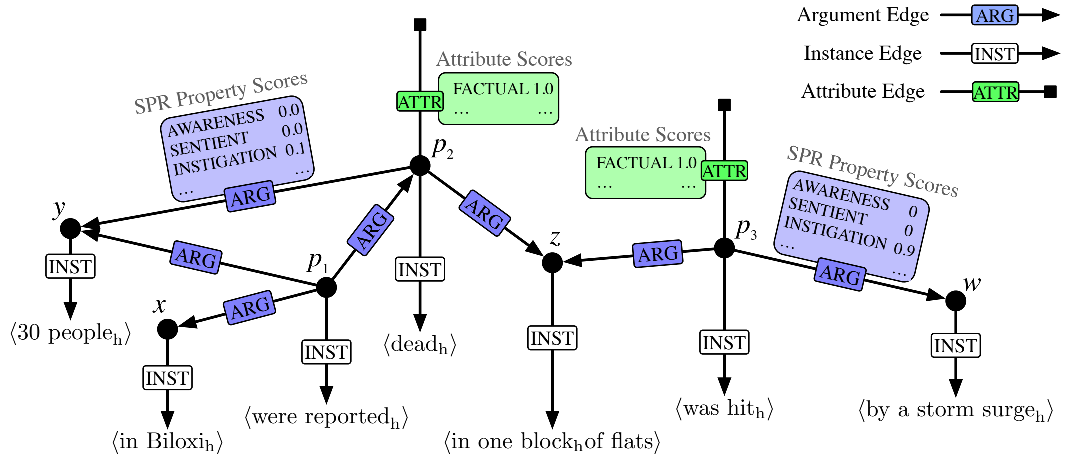
Many of these efforts are limited to the analysis of English, but with a number of exceptions, e.g., recent efforts by Bos et al. (2017), ongoing efforts in Minimal Recursion Semantics (MRS, Copestake et al., 1995), multilingual FrameNet annotation and parsing (Fung and Chen, 2004; Padó and Lapata, 2005), among others. For many languages, semantic analysis cannot be performed directly, owing to a lack of training data. While there is active work in the community focused on rapid construction of resources for low resource languages (Strassel and Tracey, 2016), it remains an expensive and perhaps infeasible solution to assume in-language annotated resources for developing semantic parsing technologies. In contrast, bitext is easier to get: it occurs often without researcher involvement,[1] and even when not available, it may be easier to find bilingual speakers that can translate a text, than it is to find experts that will create in-language semantic annotations. In addition, we are simply further along in being able to automatically understand English than we are other languages, resulting from the bias in investment in English-rooted resources.

Therefore, we propose the task of cross-lingual decompositional semantic parsing, which aims at transducing a sentence in the source language (e.g., Chinese sentence in Figure 4.1b) into a decompositional semantic analysis derived based on English, via bitext. The efforts of decompositional semantics (White et al., 2016) focus on approaches to annotating meaning based on fine-grained scalar judgments which reflect the ambiguity of language, and the underspecification of meaning in context. Our contributions include:

(1) A form of decompositional semantic analysis allowing systems to target varying levels of structural complexity.

(2) An evaluation metric to measure the similarity between system and reference semantic analysis.

[1]For example, owing to a government decree.



(a) UDS graph representation.

据	报道	，	在	比洛克西	，	三十	人	死	于	一	栋	被	风暴	潮	袭击	的	公寓楼
were	reported	,	in	Biloxi	,	30	people	dead	in	one	block	PTCP	storm	surge	hit	which	flats

"*30 people were reported dead in one block of flats which was hit by a storm surge*."

(b) Chinese sentence with Leipzig gloss.

Figure 4.1: Input and output of cross-lingual decompositional semantic parsing.

(3) An encoder-decoder model for cross-lingual decompositional semantic parsing. With a coreference annotating mechanism, the model solves intra-sentential coreference explicitly.

(4) The first evaluation dataset for cross-lingual decompositional semantic parsing.[2]

Experiments demonstrate our model achieves 38.78% F_1, outperforming strong baselines.

[2]http://decomp.io

4.2 Semantic Analysis

The goal of cross-lingual decompositional semantic parsing is to provide a semantic analysis which can be used for various types of deep and shallow processing on the target language side. Many forms of semantic analysis are potentially suitable for this goal, e.g., AMR (Banarescu et al., 2013), UCCA (Abend and Rappoport, 2013), and Universal Decompositional Semantics (White et al., 2016). Here we choose Universal Decompositional Semantics (UDS), but note that our approach is applicable to other potential graph semantic formalisms.

The reasons for choosing UDS are three-fold: (1) **Compatibility**: UDS relates to Robust Minimal Recursion Semantics (RMRS, Copestake, 2007), aiming for a maximal degree of semantic compatibility. With UDS, shallow analysis, such as predicate-argument extraction in Chapter 3, can be regarded as producing a semantics which is underspecified and reusable with respect to deeper analysis, such as lexical semantics and inference (White et al., 2016). (2) **Robustness and Speed**: There exists a robust framework, PredPatt (White et al., 2016), for automatically creating UDS from raw sentences and their Universal Dependencies. In § 2.2, we have shown that PredPatt is fast and accurate enough to process large volumes of text. (3) **Cross-lingual validity**: PredPatt is based purely on non-lexical and linguistically well-founded patterns from Universal Dependencies, which is designed to be cross-linguistically consistent.

There are three forms to represent UDS: flat, graph, or linearized representations. They are created for different purposes, and are inter-convertible.

4.2.1 Graph Representation

The graph representation as shown in Figure 4.1a is developed to improve ease of readability, parser evaluation, and integration with lexical semantics. The structure of the graph representation is a tuple $\mathcal{G} = \langle V, E \rangle$: a set of variables V (e.g., p_1 and x), and a set of edges E. There are 3 types of edges: (1) Argument edges describe argument relations between variable pairs. Deeper analysis such as Semantic Proto-Role (SPR) properties (Reisinger et al., 2015) can be attached to argument edges. SPR analysis can be considered as a scalar regression problem (White et al., 2016), where each predicate-argument pair is annotated with scalar values for different SPR properties. (2) Instance edges describe instances of variables in the target language. The subscript "h" indicates the syntactic head of an instance. (3) Attribute edges are unary, which describe various attributes of variables, such as event factuality (Saurí and Pustejovsky, 2009) and word senses (Miller, 1995). The graph representation can be viewed as an underspecified version of Dependency Minimal Recursion Semantics (DMRS) (Copestake, 2009) due to the underspecification of scope. Different from DMRS, the graph representation is linked cleanly to Universal Dependency syntax via PredPatt.

4.2.2 Linearized Representation

The linearized representation aims to facilitate learning of semantic parsers. Recently parsers based on RNN that make use of linearized representation have achieved state-of-the-art performance in constituency parsing (Choe and Charniak, 2016), logical form prediction (Dong and Lapata, 2016; Jia and Liang, 2016), and AMR parsing (Barzdins and Gosko, 2016b; Peng et al., 2017b). There was also work on predicting linearized semantic representations before RNN based approaches (Wong and Mooney, 2006).

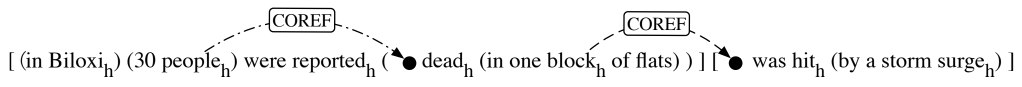

COREF
COREF
[(in Biloxi_h) (30 people_h) were reported_h (● dead_h (in one block_h of flats))] [● was hit_h (by a storm surge_h)]

Figure 4.2: UDS linearized representation. Deeper analysis such as SPR and factuality is not shown.

Figure 4.2 shows an example of UDS linearized representation. Intra-sentential coreference occurs when an instance refers to an antecedent, where we replace the instance with a special symbol "•" and add a COREF link between "•" and its antecedent. The linearized representation can be viewed as a sequence of tokens with a list of COREF links. Brackets, parentheses, and the special symbol "•" are all considered as tokens in this representation. The COREF links are drawn as a visual convenience, and the actual linearized representation achieves this via co-indexing, and is thus fully linear. We describe the procedure of converting graph representation to linearized representation as follows: Starting at the root node of the dependency tree (i.e., "reported$_\text{h}$" in Figure 4.1), we take an in-order traversal of its spanning tree. As the tree is expanded, brackets are inserted to denote the beginning or end of a predicate span, and parentheses are inserted to denote the beginning or end of an argument span.

4.2.3 Flat Representation

The non-recursive or "flat" representation can be viewed as a Parson-style (Parsons, 1990) and underspecified version of neo-Davidsonianized RMRS (Copestake, 2007). As shown in Figure 4.3, the flat representation is a tuple $\mathcal{F} = \langle P, A \rangle$ where P is a bag of predicates that are all maximally unary, and A is a bag of arguments represented by separate binary relations.

Predicates:
$\langle \text{were reported}_{\text{h}} \rangle(p_1), \langle \text{dead}_{\text{h}} \rangle(p_2),$
$\langle \text{was hit}_{\text{h}} \rangle(p_3),$
$\langle \text{in Biloxi}_{\text{h}} \rangle(x), \langle \text{30 people}_{\text{h}} \rangle(y),$
$\langle \text{in one block}_{\text{h}} \text{ of flats} \rangle(z), \langle \text{by a storm surge}_{\text{h}} \rangle(w)$

Argument Relations:
$\text{ARG}(p_1, x), \text{ARG}(p_1, y), \text{ARG}(p_1, p_2),$
$\text{ARG}(p_2, y), \text{ARG}(p_2, z),$
$\text{ARG}(p_3, z), \text{ARG}(p_3, w),$

Figure 4.3: UDS "flat" representation. Deeper analysis such as SPR and factuality is not shown.

Predicate: Predicates in PredPatt representation are referred as *complex predicates*: they are open-class predicates represented in the target language. Scope and lexical information in the predicates are left unresolved, yet can be recovered incrementally in deep semantic parsing. From the perspective of RMRS, complex predicates are conjunctions of underspecified *elementary predications* (Copestake et al., 2005) where handles are ignored, but syntax properties from Universal Dependencies are retained. For instance, in Figure 4.3, the subscript "h" in the predicate "$\langle \text{were reported}_{\text{h}} \rangle$" indicates that "reported" is a syntactic head in the predicate.

Argument Relation: The Parson-style flat representation makes arguments first-class predications $\text{ARG}(\cdot, \cdot)$. Using this style allows incremental addition of arguments, which is useful in shallow semantics where the arity of open-class predicate and the argument indexation are underspecified. They can be recovered when lexicon is available in deep analysis (Dowty, 1989; Copestake, 2007).

4.3 Related Work

Our work synthesizes two strands of research, semantic analysis and cross-lingual learning.

The semantic analysis targeted in this work is akin to that of Hobbs (2003), but our eventual goal is to transduce texts from arbitrary human languages into a *"...broad, language-like, inference-enabling [semantic representation] in the spirit of Montague..."* (Schubert, 2015). Unlike efforts such as by Schubert and colleagues that directly target such an analysis, we are pursuing a strategy that incrementally increases the complexity of the target analysis in accordance with our ability to fashion models capable of producing it.[3] Embracing underspecification in the name of tractability is exemplified by MRS (Copestake et al., 2005; Copestake, 2009), the so-called *slacker semantics*, and we draw inspiration from that work. Analyses such as AMR (Banarescu et al., 2013) also make use of underspecification, but usually this is only implicit: certain

[3]E.g., in Figure 4.1a we recognize "*by a storm surge*" as an initial structural unit, with multiple potential analysis, which may be further refined based on the capabilities of a given cross-lingual semantic parser.

aspects of meaning are simply not annotated. Unlike AMR, but akin to decisions made in PropBank (Palmer et al., 2005) (which forms the majority of the AMR ontological backbone), we target an analysis with a close correspondence to natural language syntax. Unlike interlingua (Mitamura et al., 1991; Dorr and Habash, 2002) that maps the source language into an intermediate analysis, and then maps it into the target language, we are not concerned with generating text from the semantic analysis. Substantial prior work on semantic analyses exists, including HPSG-based analyses (Copestake et al., 2005), CCG-based analyses (Steedman, 2000; Baldridge and Kruijff, 2002; Bos et al., 2004), and Universal Dependencies based analyses (White et al., 2016; Reddy et al., 2017). See (Schubert, 2015; Abend and Rappoport, 2017) for further discussion.

Cross-lingual learning has previously been applied to various NLP tasks. Yarowsky et al. (2001); Padó and Lapata (2009); Evang and Bos (2016); Faruqui and Kumar (2015) focused on projecting existing annotations on source-language text to the target language. Zeman and Resnik (2008); Ganchev et al. (2009); McDonald et al. (2011); Naseem et al. (2012); Wang and Manning (2014) enabled model transfer by sharing features or model parameters for different languages. Sudo et al. (2004); Zhang et al. (2017b); Zhang et al. (2017c); Mei et al. (2018) worked on cross-lingual information extraction and demonstrated the advantages of end-to-end learning. In this work, we explore end-to-end cross-lingual learning.

4.4 Evaluation Metric S

UDS can be represented in three forms. Evaluating such forms is crucial to the development of parsing algorithms. However, there is no method directly available for evaluation. Related methods come from semantic parsing, whose results are mainly evaluated in three ways: (1) task correctness (Tang and Mooney, 2001), which evaluates on a specific NLP task that uses the parsing results; (2) whole-parse correctness (Zettlemoyer and Collins, 2005), which counts the number of parsing results that are completely correct; and (3) Smatch (Cai and Knight, 2013), which computes the number of exactly matched edges between two semantic structures.

Nevertheless, our task needs an evaluation metric that can be used regardless of specific tasks or domains, and is able to differentiate two UDS graph representations with similar instances, SPR analysis, or attributes. We design an evaluation metric $\boldsymbol{S}$ that computes the similarity between two graph representations.

As described in § 4.2.1, the graph representation is a tuple $\mathcal{G} = (V, E)$. For two graphs $\mathcal{G}_1 = (V_1, E_1)$ and $\mathcal{G}_2 = (V_2, E_2)$, we define the score $\boldsymbol{S}$ as the maximum soft edge matching score between $\mathcal{G}_1$ and $\mathcal{G}_2$:

$$\boldsymbol{S}(\mathcal{G}_1, \mathcal{G}_2) = \max_{m \in \mathcal{M}} \Big[\sum_{(e_1^{(i)}, e_2^{(j)}) \in \mathcal{P}} f_T(e_1^{(i)}, e_2^{(j)}) \Big]$$

where m is a mapping from variables in V_1 to variables in V_2. Given a mapping m, $\mathcal{P}$ is a set of edge pairs: for each pair $(e_1^{(i)}, e_2^{(j)})$, variables(s) in $e_1^{(i)}$ are mapped to variables(s) in $e_2^{(j)}$ via m. f_T computes the matching score for a pair of edges belonging to type $T \in \{\text{ARG, INST, ATTR}\}$. The matching score is normalized to $[0, 1]$.

The precision and recall are computed by $\boldsymbol{S}(\mathcal{G}_1, \mathcal{G}_2)/|E_1|$ and $\boldsymbol{S}(\mathcal{G}_1, \mathcal{G}_2)/|E_2|$ respectively.

In this work, $f_{\text{ARG}} = f_{\text{ATTR}} = e^{-\text{MAE}}$, where MAE computes the mean absolute error between two set of scores s_1 and s_2: $\sum_i^n |s_1^{(i)} - s_2^{(i)}|/n$. $f_{\text{INST}} =$ BLEU (Papineni et al., 2002) which compute the BLEU score of an instance pair. Future work could consider, e.g., a modified BLEU that considers Levenshtein distance between tokens for a more robust partial-scoring in the face of transliteration errors.

Finding an optimal variable mapping m that yields the highest $\boldsymbol{S}$ is NP-complete. We instead adopt a strategy used in Smatch (Cai and Knight, 2013) that does a hill-climbing search with smart initialization plus 4 random restarts, and has been shown to give the best trade-off between accuracy and speed. Smatch for evaluating semantic structures can be considered as a special case of $\boldsymbol{S}$, where $f_T = \delta$, the Kronecker delta.

4.5 Model

We formulate the task of cross-lingual decompositional semantic parsing as a joint problem of sequence-to-sequence learning, coreference resolution and decompositional semantic analysis. The input is a sentence X in the source language, e.g., the Chinese sentence in Figure 4.1b. The output is a UDS linearized representation (Y, C, D) based on the target language: Y is a sequence of tokens; C is a set of COREF links; and D is a set of scores for decompositional analysis, such as SPR and factuality.

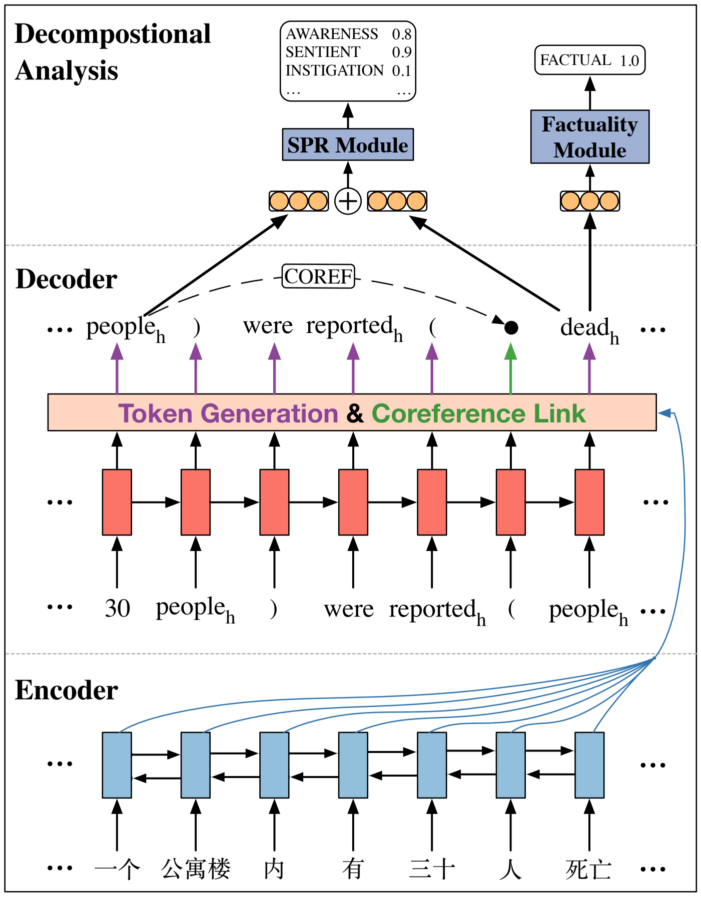
The goal is to learn a conditional probability distribution $P(Y, C, D|X)$ whose most likely configuration, given the input sentence, outputs the true UDS linearized representation with decompositional analysis. While the standard encoder-decoder framework shows the state-of-the-art performance in sequence-to-sequence learning (Choe and Charniak, 2016; Jia and Liang, 2016; Barzdins and Gosko, 2016b), it cannot directly solve intra-sentential conference and decompositional semantic analyses in our task. To achieve this goal, we propose an encoder-decoder architecture incorporated with a *coreference annotating* mechanism[4] and *decompositional analysis*. As illustrated in Figure 4.4, **Encoder** transforms the input sequence into hidden states; **Decoder** reads the hidden states, and then at each time step generates a token and creates its COREF link; **Decompositional Analysis**, based on the decoder output, performs SPR analysis for predicate-argument pairs, and factuality analysis for predicates.

4.5.1 Encoder

The encoder employs a bidirectional recurrent neural network (Schuster and Paliwal, 1997) with LSTM units (Hochreiter and Schmidhuber, 1997). It encodes the input $X = x_1, \ldots, x_N$[5] into a sequence of hidden states $\boldsymbol{h} = h_1, \ldots, h_N$. Each hidden state h_i is a concatenation of a left-to-right hidden state $\overrightarrow{h_i}$ and a right-to-left hidden state $\overleftarrow{h_i}$,

[4]Similar coreference mechanism has been proposed by Ji et al. (2017).

[5]For simplicity, we use X (and Y) to represent both tokens as well as their word embeddings.



Decompostional Analysis
AWARENESS 0.8
SENTIENT 0.9
INSTIGATION 0.1
FACTUAL 1.0
SPR Module
Factuality Module
Decoder
COREF
… people_h) were reported_h (dead_h …
Token Generation & Coreference Link
… 30 people_h) were reported_h (people_h …
Encoder
… 一个 公寓楼 内 有 三十 人 死亡 …

Figure 4.4: Illustration of the model architecture.

4.5.2 Decoder

Given the encoder hidden states, the decoder predicts the linearized representation (as shown in Figure 4.2) according to the conditional probability $P(Y, C \mid X)$ which is decomposed as a product of the decoding probabilities at each time step t:

$$P(Y, C \mid X) = \prod_{t=1}^{M} P(y_t, c_t \mid y_{<t}, c_{<t}, X) \tag{4.1}$$

where y_t is the decoded token at time step t, and c_t is the source of the COREF link for y_t, i.e., the antecedent of y_t. The set of possible antecedents of y_t is $\mathcal{A}(t) = \{\epsilon, y_1, \ldots, y_{t-1}\}$: a dummy antecedent ϵ and all preceding tokens. ϵ represents a scenario, where the token is not a special symbol "•", and it refers to none of the preceding tokens. $y_{<t}$ and $c_{<t}$ are the preceding tokens and their antecedents. We omit $y_{<t}$ and $c_{<t}$ from the notation when the context is unambiguous.

The decoding probability at each time step t is decomposed as

$$P(y_t, c_t) = P(y_t)P(c_t|y_t) \tag{4.2}$$

where $P(y_t)$ is the token generation probability, and $P(c_t|y_t)$ is the antecedent probability.

Token Generation: The probability distribution of the generated token y_t is defined as

$$P(y_t) = \text{softmax}(\text{FFNN}_\text{g}(s_t, a_t)) \tag{4.3}$$

where FFNN_g is a two-layer feed-forward neural network over the decoder hidden state s_t and the attention-weighted vector a_t. s_t is computed by

$$s_t = \text{RNN}(y_{t-1}, s_{t-1}), \tag{4.4}$$

where RNN is a recurrent neural network using LSTM. a_t is computed by the attention mechanism (Bahdanau et al., 2014; Luong et al., 2015),

$$a_t = \sum_i^N \alpha_{t,i} h_i, \tag{4.5}$$

$$\alpha_{t,i} = \frac{\exp\left(s_t^\top (W_a h_i + b_a)\right))}{\sum_{j=1}^N \exp\left(s_t^\top (W_a h_j + b_a)\right)}, \tag{4.6}$$

where W_a is a transform matrix and b_a is a bias.

Coref Link: The probability of y_t referring to the preceding token y_k, i.e., $c_t = y_k$, is defined as

$$P(c_t = y_k | y_t) = \frac{\exp\left(\textsc{score}(y_t, y_k)\right)}{\sum_{y_k' \in \mathcal{A}(t)} \exp\left(\textsc{score}(y_t, y_{k'})\right)}, \tag{4.7}$$

$\textsc{score}(y_t, y_k)$ is a pairwise score for a COREF link from y_k to y_t, defined as:

$$\textsc{score}(y_t, y_k) = s_c(y_t) + s_p(y_k) + s_a(y_t, y_k) \tag{4.8}$$

There are three factors in this pairwise score, which is akin to Lee et al. (2017): (1) $s_c(y_t)$, whether y_t should refer to a preceding instance; (2) $s_p(y_k)$, whether y_k shoud be a candidate source of such a coreference; and (3) $s_a(y_t, y_k)$, whether y_k is an antecedent of y_t.

Figure 4.5 shows the details of the scoring architecture. At the core of the three factors are vector representations $\gamma(y_t)$ for each token y_t, which is described in detail in the following section. Given the currently considered token y_t and a preceding token y_k, the scoring functions above are computed

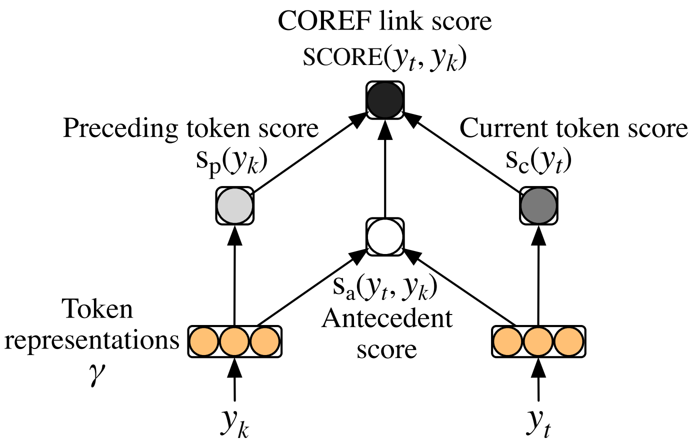


Figure 4.5: Scoring architecture in the copy mechanism between a preceding token y_k and the currently considered token y_t.

via standard feed-foward neural networks:

$$s_c(y_t) = w_c \cdot \textsc{ffnn}_c(\gamma(y_t)) \tag{4.9}$$

$$s_p(y_k) = w_p \cdot \textsc{ffnn}_p(\gamma(y_k)) \tag{4.10}$$

$$\begin{aligned} s_a(y_t, y_k) = & w_a \cdot \textsc{ffnn}_a([\gamma(y_t), \gamma(y_k), \\ & \gamma(y_t) \circ \gamma(y_k)]) \end{aligned} \tag{4.11}$$

where $\cdot$ denotes dot product, $\circ$ denotes element-wise multiplication, and FFNN denotes a two-layer feed-foward neural network over the input. The input of $\textsc{ffnn}_a$ is a concatenation of vector representations $\gamma(y_t)$ and $\gamma(y_k)$, and their explicit element-wise similarity $\gamma(y_t) \circ \gamma(y_k)$.

Token representations: To accurately predict COREF link scores as well as decompositional analysis (which is described in the following section), we consider three types of information in each token representation $\gamma(y_t)$: (1) the token itself y_t, (2) on the decoder side, the preceding context $y_{<t}$, and (3) on the encoder side, the input sequence $X = x_1, \ldots, x_N$.

The lexical information of the token itself y_t is represented by its word embedding e_t. The preceding context $y_{<t}$ is encoded by the decoder RNN in Equation (4.4). We use the decoder hidden state s_t to represent the preceding context information. The encoder-side context is represented by an attention-weighted weight a_t defined in Equation (4.6). All the above information is concatenated to produce the final token representation $\gamma(y_t)$:

$$\gamma(y_t) = [e_t, s_t, a_t] \tag{4.12}$$

4.5.3 Decompositional Analyses

The decompositional analyses D contains scores for Semantic Proto-Role (SPR) properties D_{SPR}, and scores for event factuality D_{FACT}.

SPR: Given a predicate-argument pair (y_i, y_j), we denote the score for SPR property p as $D^{(y_i,y_j)}_{\text{SPR}_p}$. As shown in Figure 4.4, we concatenate the token representations of predicate and argument head tokens $\gamma(y_i)$ and $\gamma(y_j)$ as the input to a SPR module. We employ the state-of-the-art SPR module in Rudinger et al. (2018a), defined as:

$$\hat{D}^{(y_i,y_j)}_{\text{SPR}_p} = W_{\text{SPR}_p}\text{ReLU}(W_{\text{shared}}[\gamma(y_i), \gamma(y_j)]) \tag{4.13}$$

where $\mathbf{W}_{\text{shared}}$ is the weight matrix shared across all properties. $\mathbf{W}_{\text{SPR}_p}$ is the weight matrix for SPR property p. Then, the log-likelihood of the score of SPR property p is defined as the negative L_2 loss, i.e., $-|\hat{D}^{(y_i,y_j)}_{\text{SPR}_p} - D^{(y_i,y_j)}_{\text{SPR}_p}|^2$.

Factuality: We consider predicting event factuality as a scalar regression problem (White et al., 2016), and denote the factuality score of predicate y_k as $D^{(y_k)}_{\text{FACT}}$. As shown in Figure 4.4, we take the token representation of predicate head token $\gamma(y_k)$ as the input to the state-of-the-art factuality module (Rudinger et al., 2018b):

$$\hat{D}^{(y_k)}_{\text{FACT}} = V_2 \text{ReLU}\left(V_1 \gamma(y_k) + b_1\right) + b_2, \tag{4.14}$$

where V_1 and V_2 are weight matrices, and b_1 and b_2 are biases. The log-likelihood of factuality score is defined as negative of the Huber loss (Huber, 1964) with $\delta = 1$.

We assume conditional independence among decompositional analysis:

$$\begin{aligned} P(D|X,Y,C) = \prod_{(y_i,y_j)} \prod_{p} (D^{(y_i,y_j)}_{\text{SPR}_p} | X, Y, C) \\ \prod_{y_k} P(D^{(y_k)}_{\text{FACT}} | X, Y, C) \end{aligned} \tag{4.15}$$

4.5.4 Learning

Given the input sentence X, the output sequence of tokens Y, and the COREF links C, and the decompositional analysis D, the objective is to minimize the below negative log-likelihood:

$$\begin{aligned} \mathcal{L} &= -\log P(Y, C, D|X) \\ &= -\sum_{t=1}^{M} [\mu_1 \log P(y_t) + \mu_2 \log P(c_t|y_t)] - \\ &\quad \mu_3 \log P(D|X, Y, C) \end{aligned}$$

To increase the convergence rate, we pretrain the model by setting the weights $\mu_1 = 1$ and $\mu_2 = \mu_3 = 0$ to only optimize the token generation accuracy. After the model converges, we set $\mu_2 = \mu_3 = 1$ and lower $\mu_1 = 0.1$.

4.6 Experiments

We now describe the evaluation data, baselines, and experimental results.

4.6.1 Hyperparameters

Encoder: Word embeddings are randomly initialized 300d vectors sampled from $\mathcal{U}(-0.1, 0.1)$. The encoder RNN uses 2-layer bidirectional LSTMs with hidden state size of 500 and dropout rate at 0.3. Hidden states are zero initialized. All other parameters are sampled from $\mathcal{U}(-0.1, 0.1)$.

Decoder: Word embeddings are initialized by open-source GloVe vectors (Pennington et al., 2014) trained on Common Crawl 840B with 300 dimensions. The decoder RNN uses 2-layer LSTMs with hidden state size of 500 and dropout rate at 0.3. Hidden states are initialized by the last left-to-right hidden states of encoder. All other parameters are sampled from $\mathcal{U}(-0.1, 0.1)$.

Token Generation: The feed-forward neural network is defined as

$$\text{FFNN}_\text{g}(s_t, c_t) = \tanh(W_\text{g} \begin{bmatrix} s_t \\ c_t \end{bmatrix} + b_\text{g}) \tag{4.16}$$

All transform matrices and bias used in generation are all sampled from $\mathcal{U}(-0.1, 0.1)$.

Coref Link: All feed-forward neural networks in the coreference annotating mechanism are defined as

$$\text{FFNN}(\boldsymbol{x}) = W_3\text{ReLU}(W_2\text{ReLU}(W_1\boldsymbol{x} + b_1) + b_2) + b_3$$

where the sizes of W_1, W_2 and W_3 are 1000×500, 500×500 and 500×1 respectively. Dropout at rate of 0.3 is applied to the output of each layer. All transform matrices and bias used in the copying mechanism are all sampled from $\mathcal{U}(-0.1, 0.1)$.

SPR module: The SPR model is a two-layer perceptron:

$$\hat{D}^{(y_i,y_j)}_{\text{SPR}_p} = W_{\text{SPR}_p}\text{ReLU}(W_{\text{shared}}[\gamma(y_i), \gamma(y_j)]) \tag{4.17}$$

where size of W_{shared} is 2648×2648 and sizes of all W_{SPR_p} are 2648×1. All transform matrices used in the SPR model are all sampled from $\mathcal{U}(-0.1, 0.1)$.

Factuality module: The Factuality model is a two-layer perceptron:

$$\hat{D}^{(y_k)}_{\text{FACT}} = V_2\text{ReLU}\left(V_1\gamma(y_k) + b_1\right) + b_2, \tag{4.18}$$

where size of $V1$ and $V2$ are 1324×1324 and 1324×1. All transform matrices used in the factuality model are all sampled from $\mathcal{U}(-0.1, 0.1)$.

4.6.2 Data

We choose Chinese as the source language and English as the target language. For **test**, we select 270 sentences from the Universal Dependencies (UD) English Treebank (Silveira et al., 2014) test set, which have human-annotated SPR (White et al., 2016) and factuality (Rudinger et al., 2018b) analyses. We then create linearized representations for these sentences using

PredPatt based on their gold UD syntax. Meanwhile, the Chinese translations of these sentences are created by crowdworkers on Amazon Mechanical Turk. For **training**, we use the dataset introduced in Chapter 3 for Chinese-English open information extraction, which contains tokenzied Chinese sentences and parallel English sentences with linearzied PredPatt representations. We add SPR and factuality annotations on the English side using the state-of-the-art models (Rudinger et al., 2018a; Rudinger et al., 2018b) trained on SPR v2.x and It-happened v2.0 respectively.[6] We hold out 20K training sentences for **validation** and **in-domain test**. Table 4.1 reports the dataset statistics.

	No. sents	Source
Train	1,879,172	GALE
Validation	10,000	GALE
In-domain Test	10,000	GALE
Test	270	UD Treebank

Table 4.1: Statistics of the evaluation data.

4.6.3 Variants

We evaluate our model described in § 4.5 and three variants: **(a)** We replace the coreference annotating mechanism by randomly choosing an antecedent from all preceding instances. **(b)** We preprocess the data by replacing the special symbol "•" with the syntactic head of its antecedent. During training and testing, we replace the coreference annotating mechanism with a heuristic that solves coreference by randomly choosing an antecedent among preceding

[6]Both datasets are available at `http://decomp.net`

instances which have the same syntactic head. **(c)** We remove the decoder-side information in the token representation $\gamma(y_t)$ defined in Equation (4.12) and only keep the encoder-side information a_t. We also include a **Pipeline** approach where Chinese sentences are first translated into English by a neural machine translation system (Klein et al., 2017) and are then annotated by a UD parser (Andor et al., 2016). The UDS linearized representation of **Pipeline** are created by PredPatt based the automatic UD parses.

	***S* metric**			$\textbf{BLEU}_{\text{INST}}$	$\textbf{MAE}_{\text{SPR}}$	$\textbf{MAE}_{\text{FACT}}$
	Precision	Recall	F1			
Pipeline	35.08	30.10	32.39	15.03	N/A	N/A
Variant (a)	39.31	32.93	35.84	16.74	0.75	1.11
Variant (b)	42.76	33.20	37.38	17.71	0.74	1.14
Variant (c)	41.74	33.28	37.03	18.01	0.80	1.14
Our model	**45.33**	**33.88**	**38.78**	**19.61**	**0.71**	**1.06**

Table 4.2: Evaluation of results on the **test** set.

	***S* metric**			$\textbf{BLEU}_{\text{INST}}$	$\textbf{MAE}_{\text{SPR}}$	$\textbf{MAE}_{\text{FACT}}$
	Precsion	Recall	F1			
Pipeline	35.42	23.53	28.27	14.80	N/A	N/A
Variant (a)	37.46	25.91	30.63	15.67	0.45	0.77
Variant (b)	41.27	26.41	32.21	16.89	0.44	0.79
Variant (c)	40.27	26.40	31.89	16.60	0.43	**0.74**
Our model	**44.17**	**27.04**	**33.55**	**17.90**	**0.43**	0.78

Table 4.3: Evaluation of results on the **in-domain** test set.

4.6.4 Results

Table 4.2 and Table 4.3 report the experimental results on the **test** set and the **in-domain test** set. ***S* metric** (defined in § 4.4) measures the similarity between predicted and reference graph representations. Based on the optimal variable mapping provided by the ***S* metric**, we are able to evaluate our model and the variants in different aspects: $\textbf{BLEU}_{\text{INST}}$ measures the BLEU score of all matched instance edges; $\textbf{MAE}_{\text{SPR}}$ measures the mean absolute error of SPR property scores of all matched argument edges; and $\textbf{MAE}_{\text{FACT}}$ measures the mean absolute error of factuality scores of all matched attribute edges.

Overall, our proposed model outperforms the variants in every aspect. Variants (a) and (b) use simple heuristics to solve coreference, and achieve reasonable results: they both employ sequence-to-sequence models to predict graph representations, which can be considered a replica of state-of-the-art approaches for structured prediction (Choe and Charniak, 2016; Barzdins and Gosko, 2016b; Peng et al., 2017b). Compared to our model which employs the coreference annotating mechanism, these two variants suffer notable loss in the precision of S metric. As a result, their performance drops on the other metrics. Variant (c) only uses the encoder-side information for token representation, resulting in significant loss in MAE_{SPR} and MAE_{FACT}. In the pipeline approach, each component is trained independently. During test, residual errors from each component are propagated through the pipeline. As expected, it shows a significant performance drop.

Coreference occurs 589 times in the test set. To evaluate the coreference accuracy of our model, we force the decoder to generate the reference target

	Precision	Recall	F1
Variant (a)	10.38	31.23	15.58
Variant (b)	88.42	50.59	64.36
Variant (c)	84.12	35.99	50.41
Our model	**96.63**	**97.62**	**97.12**

Table 4.4: Coreference evaluation (MUC) based on forced decoding.

sequence, and only predict coreference via the copy mechanism, or its variants. In Table 4.4, we report the precision, recall, and F_1 for the standard MUC using the official coreference scorer of the CoNLL-2011/2012 shared tasks (Pradhan et al., 2014). Since coreference in our setup occurs at the sentence level, our model achieves high performance. Variant (a) randomly choosing antecedents performs poorly, whereas variant (b), which solves coreference only based on syntactic heads, achieves a relatively high score. Variant (c) demonstrates that only using encoder-side information in the coreference annotating mechanism leads a significant performance drop.

Since our model and the state-of-the-art monolingual SPR model (Rudinger et al., 2018b) use the same test set, we are able to compare the performance of our model against the monolingual model by forcing the decoder and the coreference mechanism to create the reference graph representation and only predicting the SPR property scores. Table 4.5 shows the Pearson coefficient of each SPR property. While our model only has the access to the sentence in the source language during the encoding stage,[7] the performance is comparable to the state-of-the-art monolingual model.

[7]The state-of-the-art monolingual SPR model directly encodes the sentence in the target language.

	Our Model	Monolingual SOTA
awareness	0.852	0.879
change location	0.491	0.492
change possession	0.448	0.488
changed	0.307	0.352
change state	0.362	0.373
existed after	0.426	0.478
existed before	0.602	0.618
existed during	0.336	0.358
instigation	0.597	0.59
partitive	0.317	0.359
sentient	0.849	0.88
volition	0.818	0.837
was for benefit	0.566	0.578
was used	0.268	0.203

Table 4.5: Pearson coefficient of each SPR property.

4.7 Conclusions

We introduce the task of cross-lingual decompositional semantic parsing, which maps content provided in a source language into decompositional analysis based on a target language. We present: UDS graph/linearized representations as the target semantic interface, the ***S*** metric for evaluation, and the Chinese-English decompositional semantic parsing dataset. We propose an end-to-end learning approach with a coreference annotating mechanism which outperforms three strong baselines. We separately evaluate the coreference mechanism and SPR prediction, showing promising results. The representations for cross-lingual decompositional semantics, the evaluation metric, and the evaluation dataset provided in this work will be beneficial to the increasing interests in semantic analysis and cross-lingual applications.

Part II

Broad-Coverage Semantic Parsing

Chapter 5

AMR Parsing as Sequence-to-Graph Transduction

5.1 Introduction

Abstract Meaning Representation (AMR, Banarescu et al., 2013) parsing is the task of transducing natural language text into AMR, a graph-based formalism used for capturing sentence-level semantics. Challenges in AMR parsing include: (1) its property of reentrancy – the same concept can participate in multiple relations – which leads to graphs in contrast to trees (Wang et al., 2015); (2) the lack of gold alignments between nodes (concepts) in the graph and words in the text which limits attempts to rely on explicit alignments to generate training data (Flanigan et al., 2014; Wang et al., 2015; Damonte et al., 2017; Foland and Martin, 2017; Peng et al., 2017b; Groschwitz et al., 2018; Guo and Lu, 2018); and (3) relatively limited amounts of labeled data (Konstas et al., 2017).

Recent attempts to overcome these challenges include: modeling alignments as latent variables (Lyu and Titov, 2018); leveraging external semantic

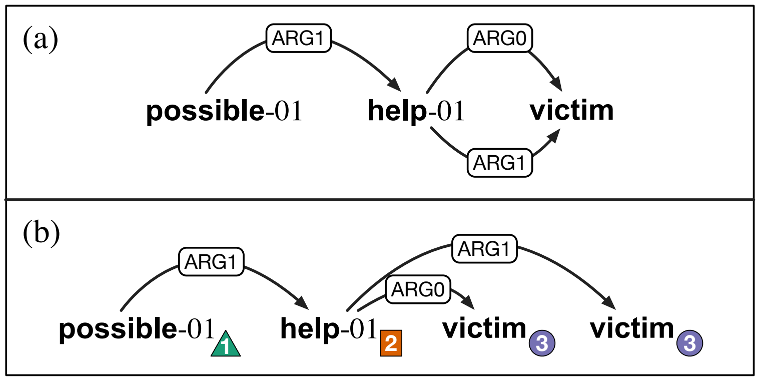


Figure 5.1: Two views of reentrancy in AMR for an example sentence *"The victim could help himself."* (a) A standard AMR graph. (b) An AMR tree with node indices as an extra layer of annotation, where the corresponding graph can be recovered by merging nodes of the same index and unioning their incoming edges.

resources (Artzi et al., 2015; Bjerva et al., 2016); data augmentation (Konstas et al., 2017; Noord and Bos, 2017b); and employing attention-based sequence-to-sequence models (Barzdins and Gosko, 2016a; Konstas et al., 2017; Noord and Bos, 2017b).

In this paper, we introduce a different way to handle reentrancy, and propose an attention-based model that treats AMR parsing as sequence-to-graph transduction. The proposed model, supported by an extended pointer-generator network, is aligner-free and can be effectively trained with limited amount of labeled AMR data. Experiments on two publicly available AMR benchmarks demonstrate that our parser clearly outperforms the previous best parsers on both benchmarks. It achieves the best reported SMATCH scores: 76.3% F1 on LDC2017T10 and 70.2% F1 on LDC2014T12. We also provide extensive ablative and qualitative studies, quantifying the contributions from each component. Our model implementation is available at `https://github.com/sheng-z/stog`.

5.2 Another View of Reentrancy

AMR is a rooted, directed, and usually acyclic graph where nodes represent concepts, and labeled directed edges represent the relationships between them (see Figure 5.1 for an AMR example). The reason for AMR being a graph instead of a tree is that it allows reentrant semantic relations. For instance, in Figure 5.1(a) "**victim**" is both `ARG0` and `ARG1` of "**help**-01". While efforts have gone into developing graph-based algorithms for AMR parsing (Chiang et al., 2013; Flanigan et al., 2014), it is more challenging to parse a sentence into an AMR graph rather than a tree as there are efficient off-the-shelf tree-based algorithms, e.g., Chu and Liu (1965); Edmonds (1968). To leverage these tree-based algorithms as well as other structured prediction paradigms (McDonald et al., 2005), we introduce another view of reentrancy.

AMR reentrancy is employed when a node participates in multiple semantic relations. We convert an AMR graph into a tree by duplicating nodes that have reentrant relations; that is, whenever a node has a reentrant relation, we make a copy of that node and use the copy to participate in the relation, thereby resulting in a tree. Next, in order to preserve the reentrancy information, we add an extra layer of annotation by assigning an index to each node. Duplicated nodes are assigned the same index as the original node. Figure 5.1(b) shows a resultant AMR tree: subscripts of nodes are indices; two "**victim**" nodes have the same index as they refer to the same concept. The original AMR graph can be recovered by merging identically indexed nodes and unioning edges from/to these nodes. Similar ideas were used by Artzi et al. (2015) who introduced Skolem IDs to represent anaphoric references in the transformation from CCG to AMR, and Noord and Bos (2017a) who kept co-indexed AMR variables, and converted them to numbers.

5.3 Task Formalization

If we consider the AMR tree with indexed nodes as the prediction target, then our approach to parsing is formalized as a two-stage process: **node prediction** and **edge prediction**.[1] An example of the parsing process is shown in Figure 5.2.

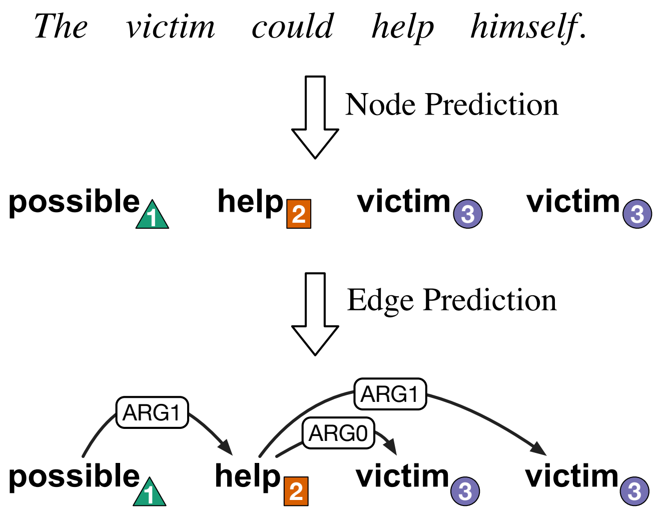


Figure 5.2: A two-stage process of AMR parsing. We remove senses (i.e., -01, -02, etc.) as they will be assigned in the post-processing step.

Node Prediction Given a input sentence $\boldsymbol{w} = \langle w_1, ..., w_n \rangle$, each w_i a word in the sentence, our approach *sequentially* decodes a list of nodes $\boldsymbol{u} = \langle u_1, ..., u_m \rangle$

[1]The two-stage process is similar to "*concept identification*" and "*relation identification*" in Flanigan et al. (2014); Zhou et al. (2016); Lyu and Titov (2018); inter alia.

and *deterministically* assigns their indices $\boldsymbol{d} = \langle d_1, ..., d_m \rangle$.

$$P(\boldsymbol{u}) = \prod_{i=1}^{m} P(u_i \mid u_{<i}, d_{<i}, \boldsymbol{w})$$

Note that we allow the same node to occur multiple times in the list; multiple occurrences of a node will be assigned the same index. We choose to predict nodes sequentially rather than simultaneously, because (1) we believe the current node generation is informative to the future node generation; (2) variants of efficient sequence-to-sequence models (Bahdanau et al., 2014; Vinyals et al., 2015a) can be employed to model this process. At the training time, we obtain the reference list of nodes and their indices using a pre-order traversal over the reference AMR tree. We also evaluate other traversal strategies, and will discuss their difference in § 5.7.2.

Edge Prediction Given a input sentence $\boldsymbol{w}$, a node list $\boldsymbol{u}$, and indices $\boldsymbol{d}$, we look for the highest scoring parse tree y in the space $\mathcal{Y}(\boldsymbol{u})$ of valid trees over $\boldsymbol{u}$ with the constraint of $\boldsymbol{d}$. A parse tree y is a set of directed head-modifier edges $y = \{(u_i, u_j) \mid 1 \leq i, j \leq m\}$. In order to make the search tractable, we follow the arc-factored graph-based approach (McDonald et al., 2005; Kiperwasser and Goldberg, 2016), decomposing the score of a tree to the sum of the score of its head-modifier edges:

$$\text{parse}(\boldsymbol{u}) = \underset{y \in \mathcal{Y}(\boldsymbol{u})}{\text{argmax}} \sum_{(u_i, u_j) \in y} \text{score}(u_i, u_j)$$

Based on the scores of the edges, the highest scoring parse tree (i.e., maximum spanning arborescence) can be efficiently found using the Chu-Liu-Edmonnds algorithm. We further incorporate indices as constraints in the

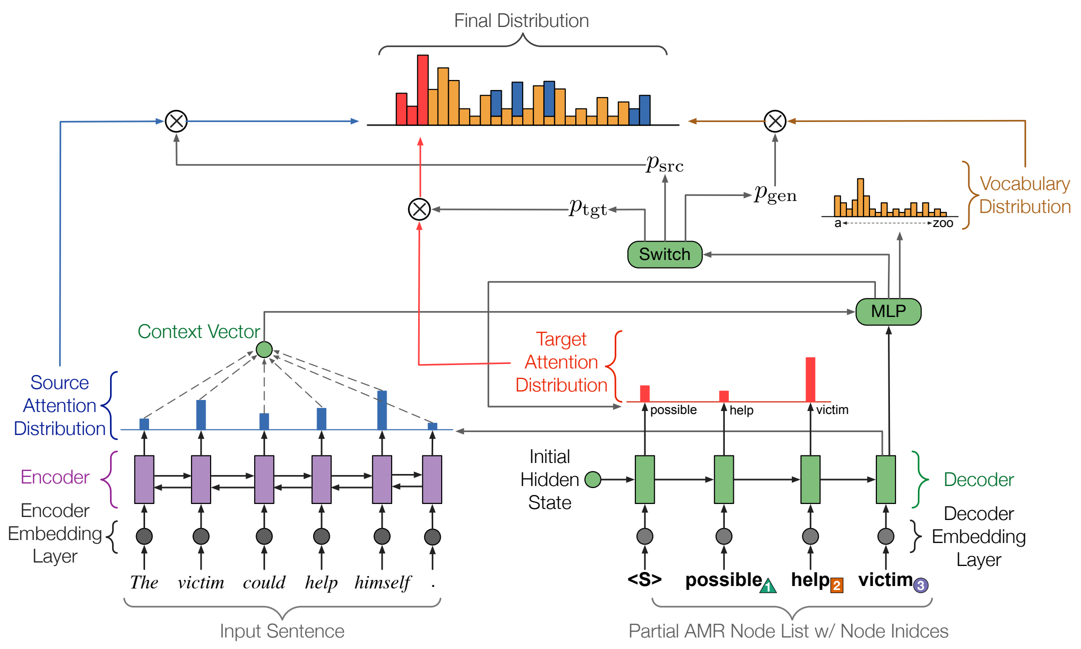


Figure 5.3: Extended pointer-generator network for node prediction. For each decoding time step, three probabilities p_{src}, p_{tgt} and p_{gen} are calculated. The source and target attention distributions as well as the vocabulary distribution are weighted by these probabilities respectively, and then summed to obtain the final distribution, from which we make our prediction. Best viewed in color.

algorithm, which is described in § 5.4.4. After obtaining the parse tree, we merge identically indexed nodes to recover the standard AMR graph.

5.4 Model

Our model has two main modules: (1) an extended pointer-generator network for node prediction; and (2) a deep biaffine classifier for edge prediction. The two modules correspond to the two-stage process for AMR parsing, and they are *jointly* learned during training.

5.4.1 Extended Pointer-Generator Network

Inspired by the *self-copy* mechanism in Zhang et al. (2018), we extend the pointer-generator network (See et al., 2017) for node prediction. The pointer-generator network was proposed for text summarization, which can copy words from the source text via *pointing*, while retaining the ability to produce novel words through the *generator*. The major difference of our extension is that it can copy nodes, not only from the source text, but also from the previously generated nodes on the target side. This *target-side pointing* is well-suited to our task as nodes we will predict can be copies of other nodes. While there are other pointer/copy networks (Gulcehre et al., 2016; Merity et al., 2016; Gu et al., 2016; Miao and Blunsom, 2016; Nallapati et al., 2016), we found the pointer-generator network very effective at reducing data sparsity in AMR parsing, which will be shown in § 5.7.2.

As depicted in Figure 5.3, the extended pointer-generator network consists of four major components: an encoder embedding layer, an encoder, a decoder embedding layer, and a decoder.

Encoder Embedding Layer This layer converts words in input sentences into vector representations. Each vector is the concatenation of embeddings of GloVe (Pennington et al., 2014), BERT (Devlin et al., 2018), POS (part-of-speech) tags and anonymization indicators, and features learned by a character-level convolutional neural network (CharCNN, Kim et al., 2016).

Anonymization indicators are binary, telling the encoder whether the word is an anonymized word. In preprocessing, text spans of named entities in input sentences will be replaced by anonymized tokens (e.g. `person`, `country`) to reduce sparsity (see § 5.6 for details).

Except BERT, all other embeddings are fetched from their corresponding learned embedding look-up tables. BERT takes subword units as input, which means that one word may correspond to multiple hidden states of BERT. In order to accurately use these hidden states to represent each word, we apply an average pooling function to the outputs of BERT. Figure 5.4 illustrates the process of generating word-level embeddings from BERT.

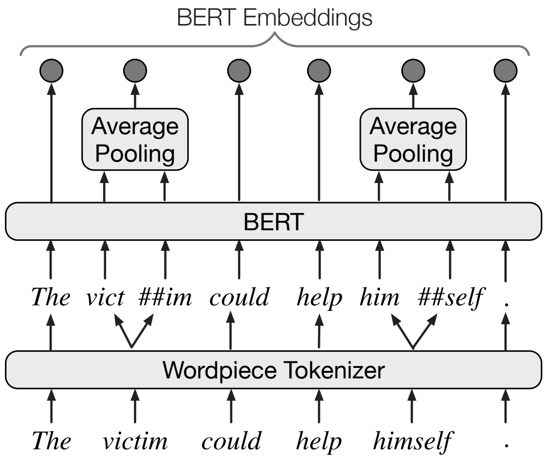

BERT Embeddings
Average Pooling
Average Pooling
BERT
The vict ##im could help him ##self .
Wordpiece Tokenizer
The victim could help himself .

Figure 5.4: Word-level embeddings from BERT.

Encoder The encoder is a multi-layer bidirectional RNN (Schuster and Paliwal, 1997):

$$\boldsymbol{h}_i^l = [\overrightarrow{f}^l(\boldsymbol{h}_i^{l-1}, \boldsymbol{h}_{i-1}^l); \overleftarrow{f}^l(\boldsymbol{h}_i^{l-1}, \boldsymbol{h}_{i+1}^l)],$$

where $\overrightarrow{f}^l$ and $\overleftarrow{f}^l$ are two LSTM cells (Hochreiter and Schmidhuber, 1997); $\boldsymbol{h}_i^l$ is the l-th layer encoder hidden state at the time step i; $\boldsymbol{h}_i^0$ is the encoder embedding layer output for word w_i.

Decoder Embedding Layer Similar to the encoder embedding layer, this layer outputs vector representations for AMR nodes. The difference is that each vector is the concatenation of embeddings of GloVe, POS tags and indices, and feature vectors from CharCNN.

POS tags of nodes are inferred at runtime: if a node is a copy from the input sentence, the POS tag of the corresponding word is used; if a node is a copy from the preceding nodes, the POS tag of its antecedent is used; if a node is a new node emitted from the vocabulary, an `UNK` tag is used.

We do not include BERT embeddings in this layer because AMR nodes, especially their order, are significantly different from natural language text (on which BERT was pre-trained). We tried to use "fixed" BERT in this layer, which did not lead to improvement.[2]

Decoder At each step t, the decoder (an l-layer unidirectional LSTM) receives hidden state $\boldsymbol{s}_t^{l-1}$ from the last layer and hidden state $\boldsymbol{s}_{t-1}^{l}$ from the previous time step, and generates hidden state $\boldsymbol{s}_t^l$:

$$\boldsymbol{s}_t^l = f^l(\boldsymbol{s}_t^{l-1}, \boldsymbol{s}_{t-1}^l),$$

where $\boldsymbol{s}_t^0$ is the concatenation (i.e., the *input-feeding* approach, Luong et al., 2015) of two vectors: the decoder embedding layer output for the previous node u_{t-1} (while training, u_{t-1} is the previous node of the reference node list; at test time it is the previous node emitted by the decoder), and the attentional vector $\widetilde{\boldsymbol{s}}_{t-1}$ from the previous step (explained later in this section). $\boldsymbol{s}_0^l$ is the concatenation of last *encoder hidden states* from $\overrightarrow{f}^l$ and $\overleftarrow{f}^l$ respectively.

[2]Limited by the GPU memory, we do not fine-tune BERT on this task and leave it for future work.

Source attention distribution $\boldsymbol{a}^t_{\text{src}}$ is calculated by additive attention (Bahdanau et al., 2014):

$$\boldsymbol{e}^t_{\text{src}} = \boldsymbol{v}^\top_{\text{src}}\tanh(\boldsymbol{W}_{\text{src}}\boldsymbol{h}^l_{1:n} + \boldsymbol{U}_{\text{src}}\boldsymbol{s}^l_t + \boldsymbol{b}_{\text{src}}),$$

$$\boldsymbol{a}^t_{\text{src}} = \text{softmax}(\boldsymbol{e}^t_{\text{src}}),$$

and it is then used to produce a weighted sum of encoder hidden states, i.e., the context vector $\boldsymbol{c}_t$.

Attentional vector $\widetilde{\boldsymbol{s}}_t$ combines both source and target side information, and it is calculated by an MLP (shown in Figure 5.3):

$$\widetilde{\boldsymbol{s}}_t = \tanh(\boldsymbol{W}_c[\boldsymbol{c}_t; \boldsymbol{s}^l_t] + \boldsymbol{b}_c)$$

The attentional vector $\widetilde{\boldsymbol{s}}_t$ has 3 usages:

(1) it is fed through a linear layer and softmax to produce the vocabulary distribution:

$$P_{\text{vocab}} = \text{softmax}(\boldsymbol{W}_{\text{vocab}}\widetilde{\boldsymbol{s}}_t + \boldsymbol{b}_{\text{vocab}})$$

(2) it is used to calculate the *target attention distribution* $\boldsymbol{a}^t_{\text{tgt}}$:

$$\boldsymbol{e}^t_{\text{tgt}} = \boldsymbol{v}^\top_{\text{tgt}}\tanh(\boldsymbol{W}_{\text{tgt}}\widetilde{\boldsymbol{s}}_{1:t-1} + \boldsymbol{U}_{\text{tgt}}\widetilde{\boldsymbol{s}}_t + \boldsymbol{b}_{\text{tgt}}),$$

$$\boldsymbol{a}^t_{\text{tgt}} = \text{softmax}(\boldsymbol{e}^t_{\text{tgt}}),$$

(3) it is used to calculate *source-side copy* probability p_{src}, *target-side copy* probability p_{tgt}, and *generation* probability p_{gen} via a *switch* layer:

$$[p_{\text{src}}, p_{\text{tgt}}, p_{\text{gen}}] = \text{softmax}(\boldsymbol{W}_{\text{switch}}\widetilde{\boldsymbol{s}}_t + \boldsymbol{b}_{\text{switch}})$$

Note that $p_{\text{src}} + p_{\text{tgt}} + p_{\text{gen}} = 1$. They act as a soft switch to choose between *copying* an existing node from the preceding nodes by sampling from the target attention distribution $\boldsymbol{a}^t_{\text{tgt}}$, or *emitting* a new node in two ways: (1) *generating* a new node from the fixed vocabulary by sampling from P_{vocab}, or (2) *copying* a word (as a new node) from the input sentence by sampling from the source attention distribution $\boldsymbol{a}^t_{\text{src}}$.

The *final probability distribution* $P^{(\text{node})}(u_t)$ for node u_t is defined as follows. If u_t is a copy of existing nodes, then:

$$P^{(\text{node})}(u_t) = p_{\text{tgt}} \sum_{i:u_i=u_t}^{t-1} \boldsymbol{a}^t_{\text{tgt}}[i],$$

otherwise:

$$P^{(\text{node})}(u_t) = p_{\text{gen}} P_{\text{vocab}}(u_t) + p_{\text{src}} \sum_{i:w_i=u_t}^{n} \boldsymbol{a}^t_{\text{src}}[i],$$

where $\boldsymbol{a}^t[i]$ indexes the i-th element of $\boldsymbol{a}^t$. Note that a new node may have the same surface form as the existing node. We track their difference using indices. The index d_t for node u_t is assigned *deterministically* as below:

$$d_t = \begin{cases} t, \text{ if } u_t \text{ is a new node;} \\ \\ d_j, \text{ if } u_t \text{ is a copy of its antecedent } u_j. \end{cases}$$

5.4.2 Deep Biaffine Classifier

For the second stage (i.e., edge prediction), we employ a deep biaffine classifier, which was originally proposed for graph-based dependency parsing (Dozat and Manning, 2016), and recently has been applied to semantic parsing (Peng et al., 2017a; Dozat and Manning, 2018).

As depicted in Figure 5.5, the major difference of our usage is that instead of re-encoding AMR nodes, we directly use *decoder hidden states* from the extended pointer-generator network as the input to deep biaffine classifier. We find two advantages of using decoder hidden states as input: (1) through the *input-feeding* approach, decoder hidden states contain contextualized information from both the input sentence and the predicted nodes; (2) because decoder hidden states are used for both node prediction and edge prediction, we can jointly train the two modules in our model.

Given decoder hidden states $\langle \boldsymbol{s}_1, ..., \boldsymbol{s}_m \rangle$ and a learnt vector representation $\boldsymbol{s}'_0$ of a dummy root, we follow Dozat and Manning (2016), factorizing edge prediction into two components: one that predicts whether or not a directed edge (u_k, u_t) exists between two nodes u_k and u_t, and another that predicts the best label for each potential edge.

Edge and label scores are calculated as below:

$$\boldsymbol{s}_t^{\text{(edge-head)}} = \text{MLP}^{\text{(edge-head)}}(\boldsymbol{s}_t)$$

$$\boldsymbol{s}_t^{\text{(edge-dep)}} = \text{MLP}^{\text{(edge-dep)}}(\boldsymbol{s}_t)$$

$$\boldsymbol{s}_t^{\text{(label-head)}} = \text{MLP}^{\text{(label-head)}}(\boldsymbol{s}_t)$$

$$\boldsymbol{s}_t^{\text{(label-dep)}} = \text{MLP}^{\text{(label-dep)}}(\boldsymbol{s}_t)$$

$$\text{score}_{k,t}^{\text{(edge)}} = \text{Biaffine}(\boldsymbol{s}_k^{\text{(edge-head)}}, \boldsymbol{s}_t^{\text{(edge-dep)}})$$

$$\text{score}_{k,t}^{\text{(label)}} = \text{Bilinear}(\boldsymbol{s}_k^{\text{(label-head)}}, \boldsymbol{s}_t^{\text{(label-dep)}})$$

where MLP, Biaffine and Bilinear are defined as below:

$$\text{MLP}(\boldsymbol{x}) = \text{ELU}(\boldsymbol{W}\boldsymbol{x} + \boldsymbol{b})$$

$$\text{Biaffine}(\boldsymbol{x}_1, \boldsymbol{x}_2) = \boldsymbol{x}_1^\top \boldsymbol{U}\boldsymbol{x}_2 + \boldsymbol{W}[\boldsymbol{x}_1; \boldsymbol{x}_2] + \boldsymbol{b}$$

$$\text{Bilinear}(\boldsymbol{x}_1, \boldsymbol{x}_2) = \boldsymbol{x}_1^\top \boldsymbol{U}\boldsymbol{x}_2 + \boldsymbol{b}$$

Given a node u_t, the probability of u_k being the edge head of u_t is defined as:

$$P_t^{\text{(head)}}(u_k) = \frac{\exp(\text{score}_{k,t}^{\text{(edge)}})}{\sum_{j=1}^{m} \exp(\text{score}_{j,t}^{\text{(edge)}})}$$

The edge label probability for edge (u_k, u_t) is defined as:

$$P_{k,t}^{\text{(label)}}(l) = \frac{\exp(\text{score}_{k,t}^{\text{(label)}}[l])}{\sum_{l'} \exp(\text{score}_{k,t}^{\text{(label)}}[l'])}$$

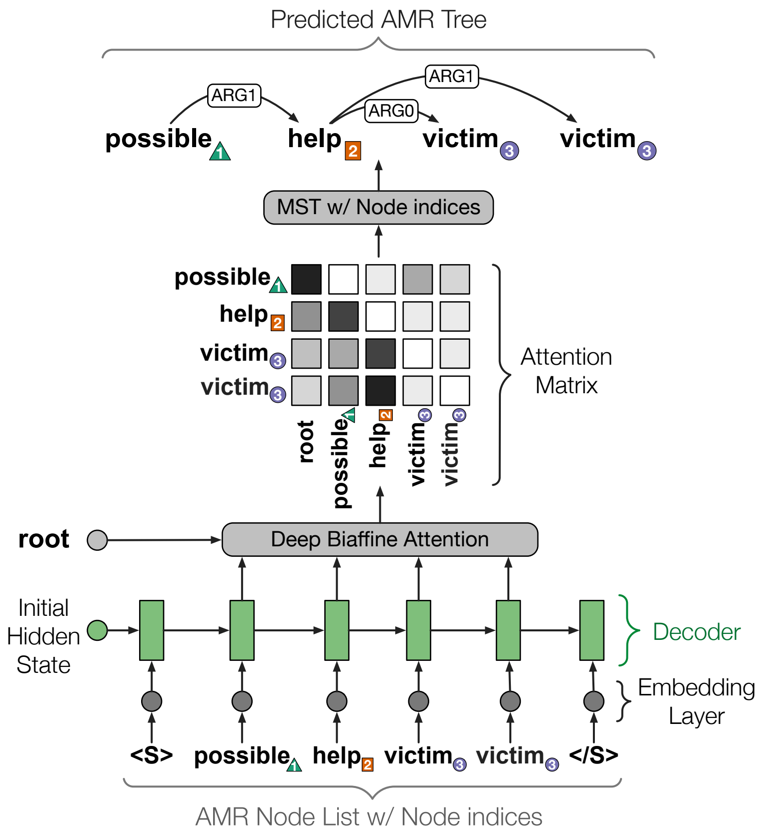


Figure 5.5: Deep biaffine classifier for edge prediction. Edge label prediction is not depicted in the figure.

5.4.3 Training

The training objective is to jointly minimize the loss of reference nodes and edges, which can be decomposed to the sum of the negative log likelihood at each time step t for (1) the reference node u_t, (2) the reference edge head u_k of node u_t, and (3) the reference edge label l between u_k and u_t:

$$\text{minimize} - \sum_{t=1}^{m} [\log P^{(\text{node})}(u_t) + \log P_t^{(\text{head})}(u_k)$$

$$+ \log P_{k,t}^{(\text{label})}(l) + \lambda \text{covloss}_t]$$

covloss_t is a *coverage loss* to penalize repetitive nodes:

$$\text{covloss}_t = \sum_i \min(\boldsymbol{a}^t_{\text{src}}[i], \mathbf{cov}^t[i]),$$

where $\mathbf{cov}^t$ is the sum of source attention distributions over all previous decoding time steps: $\mathbf{cov}^t = \sum_{t'=0}^{t-1} \boldsymbol{a}^{t'}_{\text{src}}$. See See et al. (2017) for full details.

Algorithm 1: Chu-Liu-Edmonds algo. w/ Adaptation

```
Input  : Nodes $\boldsymbol{u} = \langle u_1, ..., u_m \rangle$,
         Indices $\boldsymbol{d} = \langle d_1, ...d_m \rangle$,
         Edge scores $S = \{\text{score}^{(\text{edge})}_{i,j} \mid 0 \leq i, j \leq m\}$
Output : A maximum spanning tree.
// Include the dummy root $u_0$.
$V \leftarrow \{u_0\} \cup \boldsymbol{u}$;
$d_0 \leftarrow 0$;
// Exclude invalid edges.
// $d_i$ is the node index for node $u_i$.
$E \leftarrow \{(u_i, u_j) \mid 0 \leq i, j \leq m; d_i \neq d_j\}$;
// Chu-Liu-Edmonds algorithm
return MST($V, E, S, u_0$);
```

5.4.4 Prediction

For node prediction, based on the final probability distribution $P^{(\text{node})}(u_t)$ at each decoding time step, we implement both greedy search and beam search to sequentially decode a node list $\boldsymbol{u}$ and indices $\boldsymbol{d}$.

For edge prediction, given the predicted node list $\boldsymbol{u}$, their indices $\boldsymbol{d}$, and the edge scores $S = \{\text{score}^{(\text{edge})}_{i,j} \mid 0 \leq i, j \leq m\}$, we apply the Chu-Liu-Edmonds algorithm with a simple adaptation to find the maximum spanning tree (MST). As described in Algorithm 1, before calling the Chu-Liu-Edmonds algorithm, we first include a dummy root u_0 to ensure every node have a head, and then exclude edges whose source and destination nodes have the same indices, because these nodes will be merged into a single node to recover the standard AMR graph where self-loops are invalid.

5.5 Related Work

AMR parsing approaches can be categorized into *alignment*-based, *transition*-based, *grammar*-based, and *attention*-based approaches.

Alignment-based approaches were first explored by JAMR (Flanigan et al., 2014), a pipeline of concept and relation identification with a graph-based algorithm. Zhou et al. (2016) improved this by jointly learning concept and relation identification with an incremental model. Both approaches rely on features based on alignments. Lyu and Titov (2018) treated alignments as latent variables in a joint probabilistic model, leading to a substantial reported improvement. Our approach requires no explicit alignments, but implicitly learns a source-side copy mechanism using attention.

Transition-based approach began with Wang et al. (2015); Wang et al. (2016), who incrementally transform dependency parses into AMRs using transiton-based models, which was followed by a line of research, such as Puzikov et al. (2016); Brandt et al. (2016); Goodman et al. (2016); Damonte et al. (2017); Ballesteros and Al-Onaizan (2017); Groschwitz et al. (2018). A pre-trained aligner, e.g. Pourdamghani et al. (2014); Liu et al. (2018), is needed for most parsers to generate training data (e.g., oracles for a transition-based parser). Our approach makes no significant use of external semantic resources,[3] and is

[3]We only use POS tags in the core parsing task. In post-processing, we use an entity linker

aligner-free.

Grammar-based approaches are represented by Artzi et al. (2015); Peng et al. (2015) who leveraged external semantic resources, and employed CCG-based or SHRG-based grammar induction approaches converting logical forms into AMRs. Pust et al. (2015) recast AMR parsing as a machine translation problem, while also drawing features from external semantic resources.

Attention-based parsing with Seq2Seq-style models have been considered (Barzdins and Gosko, 2016a; Peng et al., 2017b), but are limited by the relatively small amount of labeled AMR data. Konstas et al. (2017) overcame this by making use of millions of unlabeled data through self-training, while Noord and Bos (2017b) showed significant gains via a character-level Seq2Seq model and a large amount of silver-standard AMR training data. In contrast, our approach supported by extended pointer generator can be effectively trained on the limited amount of labeled AMR data, with no data augmentation.

5.6 AMR Pre- and Post-processing

Firstly, we to run Standford CoreNLP like Lyu and Titov (2018), lemmatizing input sentences and adding POS tags to each token. Secondly, we remove senses, wiki links and polarity attributes in AMR. Thirdly, we anonymize sub-graphs of named entities and `*-entity` in a way similar to Konstas et al. (2017). Figure 5.6 shows an example before and after preprocessing. Sub-graphs of named entities are headed by one of AMR's fine-grained entity types (e.g., `highway`, `country_region` in Figure 5.6) that contain a `:name` role.

as a common move for wikification like Noord and Bos (2017b).

Sentence:
Route 288 , the circumferential highway running around the south - western quadrant of the Richmond New Urban Region , opened in late 2004 .

Anonymized Sentence:
HIGHWAY_0 , the circumferential highway running around the south - western quadrant of the COUNTRY_REGION_0 , opened in late DATE_0 .

Before preprocessing

```
(o / open-01
    :ARG1 (h / highway
        :wiki "Virginia_State_Route_288"
        :name (r / name
            :op1 "Route"
            :op2 288)
        :ARG1-of (r3 / run-04
            :direction (a / around
                :op1 (q / quadrant
                    :part-of (c / country-region
                        :wiki -
                        :name (r2 / name
                            :op1 "Richmond"
                            :op2 "New"
                            :op3 "Urban"
                            :op4 "Region"))
                    :mod (s / southwest))))
        :mod (c2 / circumference))
    :time (l / late
        :op1 (d / date-entity
            :year 2004)))
```

After preprocessing

```
(o / open
    :ARG1 (h / HIGHWAY_0
        :ARG1-of (r3 / run
            :direction (a / around
                :op1 (q / quadrant
                    :part-of (c / COUNTRY_REGION_0)
                    :mod (s / southwest))))
        :mod (c2 / circumference))
    :time (l / late
        :op1 (d / DATE_0)))
```

Figure 5.6: An example AMR and the corresponding sentence before and after preprocessing. Senses are removed. The first named entity is replaced by "HIGHWAY_0"; the second named entity is replaced by "COUNTRY_REGION_0"; the first date entity replaced by "DATE_0".

Sub-graphs of other entities are headed by their corresponding entity type name (e.g., `date-entity` in Figure 5.6). We replace these sub-graphs with a token of a special pattern "TYPE_i" (e.g. `HIGHWAY_0`, `DATE_0` in Figure 5.6), where "TYPE" indicates the AMR entity type of the corresponding sub-graph, and "i" indicates that it is the *i*-th occurrence of that type. On the training set, we use simple rules to find mappings between anonymized sub-graphs and spans of text, and then replace mapped text with the anonymized token we inserted into the AMR graph. Additionally, we build a mapping of Standford CoreNLP NER tags to AMR's fine-grained types based on the training set, which will be used in prediction. At test time, we normalize sentences to match our anonymized training data. For any entity span identified by Stanford CoreNLP, we replace it with a AMR entity type based on the mapping built during training. If no entry is found in the mapping, we replace entity spans with the coarse-grained NER tags from Stanford CoreNLP, which are also entity types in AMR.

In post-processing, we deterministically generate AMR sub-graphs for anonymizations using the corresponding text span. We assign the most frequent sense for nodes (-01, if unseen) like Lyu and Titov (2018). We add wiki links to named entities using the DBpedia Spotlight API (Daiber et al., 2013) following Bjerva et al. (2016); Noord and Bos (2017b) with the confidence threshod at 0.5. We add polarity attributes based on Algorithm 2 where the four functions `isNegation`, `modifiedWord`, `mappedNode`, and `addPolarity` consists of simple rules observed from the training set. We use the PENMAN-Codec[4] to encode and decode both intermediate and final AMRs.

Algorithm 2: Adding polarity attributes to AMR.

```
Input  : Sent. $\boldsymbol{w} = \langle w_1, ..., w_n \rangle$, Predicted AMR $A$
Output : AMR with polarity attributes.
for $w_i \in \boldsymbol{w}$ do
    if isNegation($w_i$) then
        $w_j \leftarrow$ modifiedWord($w_i$, $\boldsymbol{w}$);
        $u_k \leftarrow$ mappedNode($w_j$, $A$);
        $A \leftarrow$ addPolarity($u_k$, $A$);
    end
end
return $A$;
```

[4] https://github.com/goodmami/penman/

5.7 Experiments

GloVe.840B.300d embeddings	
dim	300
BERT embeddings	
source	BERT-Large-cased
dim	1024
POS tag embeddings	
dim	100
Anonymization indicator embeddings	
dim	50
Index embeddings	
dim	50
CharCNN	
num_filters	100
ngram_filter_sizes	[3]
Encoder	
hidden_size	512
num_layers	2
Decoder	
hidden_size	1024
num_layers	2
Deep biaffine classifier	
edge_hidden_size	256
label_hidden_size	128
Optimizer	
type	ADAM
learning_rate	0.001
max_grad_norm	5.0
Coverage loss weight λ	1.0
Beam size	5
Vocabulary	
encoder_vocab_size (AMR 2.0)	18000
decoder_vocab_size (AMR 2.0)	12200
encoder_vocab_size (AMR 1.0)	9200
decoder_vocab_size (AMR 1.0)	7300
Batch size	64

Table 5.1: Hyper-parameter settings

5.7.1 Setup

We conduct experiments on two AMR general releases (available to all LDC subscribers): AMR 2.0 (LDC2017T10) and AMR 1.0 (LDC2014T12). Our model is trained using ADAM (Kingma and Ba, 2014) for up to 120 epochs, with early stopping based on the development set. Full model training takes about 19 hours on AMR 2.0 and 7 hours on AMR 1.0, using two GeForce GTX TITAN X GPUs. At training, we have to fix BERT parameters due to the limited GPU memory. We leave fine-tuning BERT for future work.

Table 5.1 lists the hyper-parameters used in our full model. Both encoder and decoder embedding layers have GloVe and POS tag embeddings as well as CharCNN, but their parameters are not tied. We apply dropout (dropout_rate = 0.33) to the outputs of each module.

5.7.2 Results

Main Results We compare our approach against the previous best approaches and several recent competitors. Table 5.2 summarizes their SMATCH scores (Cai and Knight, 2013) on the test sets of two AMR general releases. On AMR 2.0, we outperform the latest push from Naseem et al. (2019) by 0.8% F1, and significantly improves Lyu and Titov (2018)'s results by 1.9% F1. Compared to the previous best attention-based approach (Noord and Bos, 2017b), our approach shows a substantial gain of 5.3% F1, with no usage of any silver-standard training data. On AMR 1.0 where the traininng instances are only around 10k, we improve the best reported results by 1.9% F1.

Fine-grained Results In Table 5.3, we assess the quality of each subtask using

Corpus	Parser	F1(%)
AMR 2.0	Buys and Blunsom (2017)	61.9
	Noord and Bos (2017b)	71.0*
	Groschwitz et al. (2018)	71.0±0.5
	Lyu and Titov (2018)	74.4±0.2
	Naseem et al. (2019)	75.5
	Ours	**76.3**±0.1
AMR 1.0	Flanigan et al. (2016)	66.0
	Pust et al. (2015)	67.1
	Wang and Xue (2017)	68.1
	Guo and Lu (2018)	68.3±0.4
	Ours	**70.2**±0.1

Table 5.2: SMATCH scores on the test sets of AMR 2.0 and 1.0. Standard deviation is computed over 3 runs with different random seeds. * indicates the previous best score from attention-based models.

Metric	vN'18	L'18	N'19	Ours
SMATCH	71.0	74.4	75.5	**76.3**±0.1
Unlabeled	74	77	**80**	79.0±0.1
No WSD	72	76	76	**76.8**±0.1
Reentrancies	52	52	56	**60.0**±0.1
Concepts	82	**86**	**86**	84.8±0.1
Named Ent.	79	**86**	83	77.9±0.2
Wikification	65	76	80	**85.8**±0.3
Negation	62	58	67	**75.2**±0.2
SRL	66	70	**72**	69.7±0.2

Table 5.3: Fine-grained F1 scores on the AMR 2.0 test set. vN'17 is Noord and Bos (2017b); L'18 is Lyu and Titov (2018); N'19 is Naseem et al. (2019).

the AMR-evaluation tools (Damonte et al., 2017). We see a notable increase on reentrancies, which we attribute to target-side copy (based on our ablation studies in the next section). Significant increases are also shown on wikification and negation, indicating the benefits of using DBpedia Spotlight API and negation detection rules in post-processing. On all other subtasks except named entities, our approach achieves competitive results to the previous best approaches (Lyu and Titov, 2018; Naseem et al., 2019), and outperforms the previous best attention-based approach (Noord and Bos, 2017b). The difference of scores on named entities is mainly caused by anonymization methods used in preprocessing, which suggests a potential improvement by adapting the anonymization method presented in Lyu and Titov (2018) to our approach.

Ablation	AMR 1.0	AMR 2.0
Full model	70.2	76.3
no source-side copy	62.7	70.9
no target-side copy	66.2	71.6
no coverage loss	68.5	74.5
no BERT embeddings	68.8	74.6
no index embeddings	68.5	75.5
no anonym. indicator embed.	68.9	75.6
no beam search	69.2	75.3
no POS tag embeddings	69.2	75.7
no CharCNN features	70.0	75.8
only edge prediction	88.4	90.9

Table 5.4: Ablation studies on components of our model. (Scores are sorted by the delta from the full model.)

Ablation Study We consider the contributions of several model components in Table 5.4. The largest performance drop is from removing source-side copy,[5] showing its efficiency at reducing sparsity from open-class vocabulary entries. Removing target-side copy also leads to a large drop. Specifically,

[5] All other hyper-parameter settings remain the same.

the subtask score of reentrancies drops down to 38.4% when target-side copy is disabled. Coverage loss is useful with regard to discouraging unnecessary repetitive nodes. In addition, our model benefits from input features such as language representations from BERT, index embeddings, POS tags, anonymization indicators, and character-level features from CharCNN. Note that without BERT embeddings, our model still outperforms the previous best approaches (Lyu and Titov, 2018; Guo and Lu, 2018) that are not using BERT. Beam search, commonly used in machine translation, is also helpful in our model. We provide side-by-side examples in § 5.8 to further illustrate the contribution from each component, which are largely intuitive, with the exception of BERT embeddings. There the exact contribution of the component (qualitative, before/after ablation) stands out less: future work might consider a *probing* analysis with manually constructed examples, in the spirit of Linzen et al. (2016); Conneau et al. (2018); Tenney et al. (2019).

In the last row, we only evaluate model performance at the edge prediction stage by forcing our model to decode the reference nodes at the node prediction stage. The results mean if our model could make perfect prediction at the node prediction stage, the final SMATCH score will be substantially high, which identifies node prediction as the key to future improvement of our model.

There are three sources for node prediction: vocabulary generation, source-side copy, or target-side copy. Let all reference nodes from source z be $N_{\text{ref}}^{(z)}$, and all system predicted nodes from z be $N_{\text{sys}}^{(z)}$. we compute frequency, precision and recall of nodes from source z as below:

$$\text{frequency}^{(z)} = |N_{\text{ref}}^{(z)}| \Big/ \sum_z |N_{\text{ref}}^{(z)}|$$

$$\text{precision}^{(z)} = |N_{\text{ref}}^{(z)} \cap N_{\text{sys}}^{(z)}| \Big/ |N_{\text{sys}}^{(z)}|$$

$$\text{recall}^{(z)} = |N_{\text{ref}}^{(z)} \cap N_{\text{sys}}^{(z)}| \Big/ |N_{\text{ref}}^{(z)}|$$

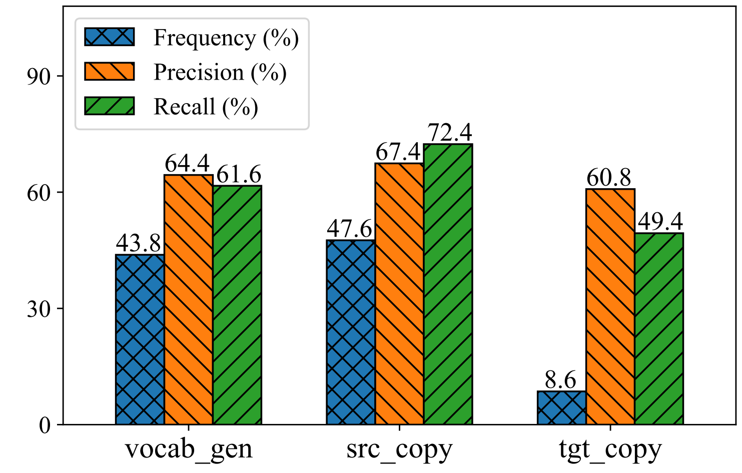


Figure 5.7: Frequency, precision and recall of nodes from different sources, based on the AMR 2.0 test set.

Figure 5.7 shows the frequency of nodes from difference sources, and their corresponding precision and recall based on our model prediction. Among all reference nodes, 43.8% are from vocabulary generation, 47.6% from source-side copy, and only 8.6% from target-side copy. On one hand, the highest frequency of source-side copy helps address sparsity and results in the highest precision and recall. On the other hand, we see space for improvement, especially on the relatively low recall of target-side copy, which is probably due to its low frequency.

Node Linearization As decribed in § 5.3, we create the reference node list by a pre-order traversal over the gold AMR tree. As for the children of each node, we sort them in alphanumerical order. This linearization strategy has two advantages: (1) pre-order traversal guarantees that a head node (*predicate*) always comes in front of its children (*arguments*); (2) alphanumerical sort orders according to role ID (i.e., `ARG0>ARG1>...>ARGn`), following intuition from research in Thematic Hierarchies (Fillmore, 1968; Levin and Hovav, 2005).

Node Linearization	AMR 1.0	AMR 2.0
Pre-order + Alphanum	70.2	76.3
Pre-order + Alignment	61.9	68.3
Pure Alignment	64.3	71.3

Table 5.5: SMATCH scores of full models trained and tested based on different node linearization strategies.

In Table 5.5, we report SMATCH scores of full models trained and tested on data generated via our linearization strategy (Pre-order + Alphanum), as compared to two obvious alternates: the first alternate still runs a pre-order traversal, but it sorts the children of each node based on the their alignments to input words; the second one linearizes nodes purely based alignments. Alignments are created using the tool by Pourdamghani et al. (2014). Clearly, our linearization strategy leads to much better results than the two alternates. We also tried other traversal strategies such as combining in-order traversal with alphanumerical sorting or alignment-based sorting, but did not get scores even comparable to the two alternates.[6]

Average Pooling vs. Max Pooling In Figure 5.4, we apply average pooling to the outputs (last-layer hidden states) of BERT in order to generate word-level embeddings for the input sentence. Table 5.6 shows scores of models using different pooling functions. Average pooling performs slightly better than max pooling.

	AMR 1.0	AMR 2.0
Average Pooling	70.2±0.1	76.3±0.1
Max Pooling	70.0±0.1	76.2±0.1

Table 5.6: SMATCH scores based different pooling functions. Standard deviation is over 3 runs on the test data.

5.8 Side-by-Side Examples

We provide examples from the test set, with side-by-side comparisons between the full model prediction and the model prediction after ablation.

Sentence:
Smoke and clouds chase the flying waves
Lemmas:
["smoke", "and", "cloud", "chase", "the", "fly", "wave"]

Full Model

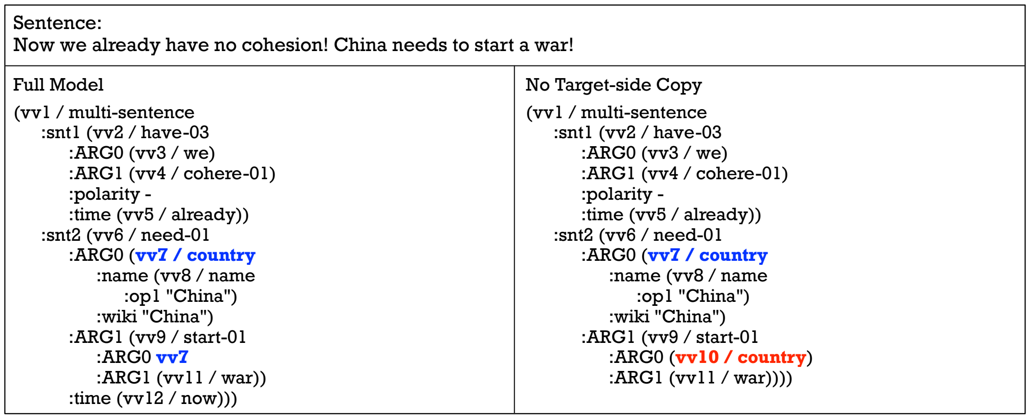
```
(vv1 / chase-01
    :ARG0 (vv2 / and
        :op1 (vv3 / smoke)
        :op2 (vv4 / cloud-01))
    :ARG1 (vv5 / wave
        :purpose (vv6 / fly-01)))
```

No Source-side Copy

```
(vv1 / and
    :op1 (vv2 / stretch-01
        :ARG1 (vv3 / and
            :op1 (vv4 / leech)))
    :op2 (vv6 / bug)
    :op3 (vv7 / fly-01)
    :op3 (vv8 / center))
```

Figure 5.8: Full model prediction vs. no source-side copy prediction. Tokens in blue are copied from the source side. Without source-side copy, the prediction becomes totally different and inaccurate in this example.

[6] Noord and Bos (2017b) also investigated linearization order, and found that alignment-based ordering yielded the best results under their setup where AMR parsing is treated as a sequence-to-sequence learning problem.



Sentence:
Now we already have no cohesion! China needs to start a war!

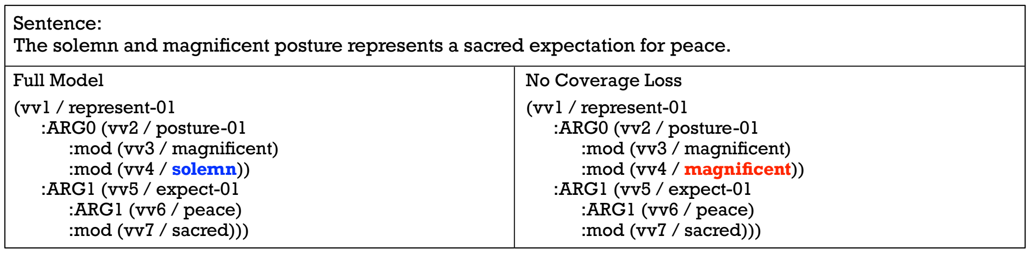
Full Model

```
(vv1 / multi-sentence
    :snt1 (vv2 / have-03
        :ARG0 (vv3 / we)
        :ARG1 (vv4 / cohere-01)
        :polarity -
        :time (vv5 / already))
    :snt2 (vv6 / need-01
        :ARG0 (vv7 / country
            :name (vv8 / name
                :op1 "China")
            :wiki "China")
        :ARG1 (vv9 / start-01
            :ARG0 vv7
            :ARG1 (vv11 / war))
        :time (vv12 / now)))
```

No Target-side Copy

```
(vv1 / multi-sentence
    :snt1 (vv2 / have-03
        :ARG0 (vv3 / we)
        :ARG1 (vv4 / cohere-01)
        :polarity -
        :time (vv5 / already))
    :snt2 (vv6 / need-01
        :ARG0 (vv7 / country
            :name (vv8 / name
                :op1 "China")
            :wiki "China")
        :ARG1 (vv9 / start-01
            :ARG0 (vv10 / country)
            :ARG1 (vv11 / war))))
```

Figure 5.9: Full model prediction vs. no target-side copy prediction. Nodes in blue denote the same concept (i.e., the country "China"). The full model correctly copies the first node ("vv7 / country") as `ARG0` of "start-01". Without target-side copy, the model has to generate a new node with a different index, i.e., "vv10 / country".



Sentence:
The solemn and magnificent posture represents a sacred expectation for peace.

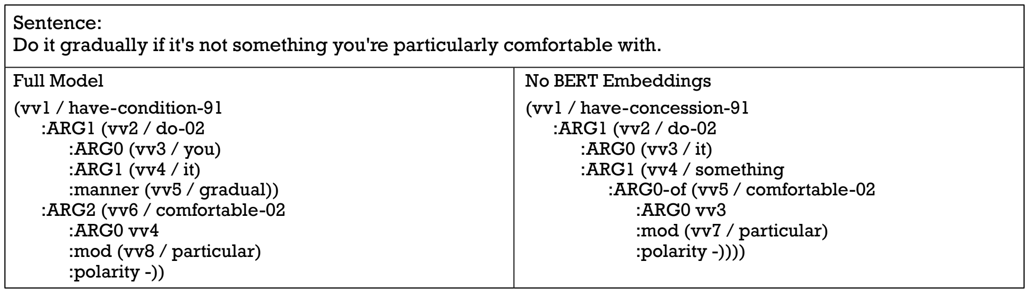
Full Model

```
(vv1 / represent-01
    :ARG0 (vv2 / posture-01
        :mod (vv3 / magnificent)
        :mod (vv4 / solemn))
    :ARG1 (vv5 / expect-01
        :ARG1 (vv6 / peace)
        :mod (vv7 / sacred)))
```

No Coverage Loss

```
(vv1 / represent-01
    :ARG0 (vv2 / posture-01
        :mod (vv3 / magnificent)
        :mod (vv4 / magnificent))
    :ARG1 (vv5 / expect-01
        :ARG1 (vv6 / peace)
        :mod (vv7 / sacred)))
```

Figure 5.10: Full model prediction vs. no coverage loss prediction. The full model correctly predicts the second modifier "solemn". Without coverage loss, the model generates a repetitive modifier "magnificent".



Sentence:
Do it gradually if it's not something you're particularly comfortable with.

Full Model

```
(vv1 / have-condition-91
    :ARG1 (vv2 / do-02
        :ARG0 (vv3 / you)
        :ARG1 (vv4 / it)
        :manner (vv5 / gradual))
    :ARG2 (vv6 / comfortable-02
        :ARG0 vv4
        :mod (vv8 / particular)
        :polarity -))
```

No BERT Embeddings

```
(vv1 / have-concession-91
    :ARG1 (vv2 / do-02
        :ARG0 (vv3 / it)
        :ARG1 (vv4 / something
            :ARG0-of (vv5 / comfortable-02
                :ARG0 vv3
                :mod (vv7 / particular)
                :polarity -))))
```

Figure 5.11: Full model prediction vs. no BERT embeddings prediction.

5.9 Conclusions

We propose an attention-based model for AMR parsing where we introduce a series of novel components into a transductive setting that extend beyond what a typical NMT system would do on this task. Our model achieves the best performance on two AMR corpora. For future work, we would like to extend our model to other semantic parsing tasks (Oepen et al., 2014; Abend and Rappoport, 2013).

Chapter 6

Broad-Coverage Semantic Parsing as Transduction

6.1 Introduction

Broad-coverage semantic parsing aims at mapping *any* natural language text, regardless of its domain, genre, or even the language itself, into a general-purpose meaning representation. As a long-standing topic of interest in computational linguistics, broad-coverage semantic parsing has targeted a number of meaning representation frameworks, including CCG (Steedman, 1996; Steedman, 2000), DRS (Kamp and Reyle, 1993; Bos, 2008), AMR (Banarescu et al., 2013), UCCA (Abend and Rappoport, 2013), SDP (Oepen et al., 2014; Oepen et al., 2015), and UDS (White et al., 2016).[1] Each of these frameworks has their specific formal and linguistic assumptions. Such framework-specific *"balkanization"* results in a variety of framework-specific parsing approaches, and the state-of-the-art semantic parser for one framework is not always

[1]Abbreviations respectively denote: Combinatory Categorical Grammar, Discourse Representation Theory, Abstract Meaning Representation, Universal Conceptual Cognitive Annotation, Semantic Dependency Parsing, and Universal Decompositional Semantics.

applicable to another. For instance, the state-of-the-art approaches to SDP parsing (Dozat and Manning, 2018; Peng et al., 2017a) are not directly transferable to AMR and UCCA because of the lack of explicit alignments between tokens in the sentence and nodes in the semantic graph.

While transition-based approaches are adaptable to different semantic parsing tasks (Wang et al., 2018; Hershcovich et al., 2018; Damonte et al., 2017), when it comes to representations such as AMR whose nodes are *unanchored* to tokens in the sentence, a pre-trained aligner has to be used to produce the reference transition sequences (Wang et al., 2015; Damonte et al., 2017; Peng et al., 2017b). In contrast, there are attempts to develop attention-based approaches in a graph-based parsing paradigm (Dozat and Manning, 2018; Zhang et al., 2019a), but they lack parsing incrementality, which is advocated in terms of computational efficiency and cognitive modeling (Nivre, 2004; Huang and Sagae, 2010).

In this work, we approach different broad-coverage semantic parsing tasks under a unified framework of transduction. We propose an attention-based neural transducer that extends the two-stage semantic parser proposed in Chapter 5 to directly transduce input text into a meaning representation in *one* stage. This transducer has properties of both transition-based approaches and graph-based approaches: on the one hand, it builds a meaning representation incrementally via a sequence of semantic relations, similar to a transition-based parser; on the other hand, it leverages multiple attention mechanisms used in recent graph-based parsers, thereby removing the need for pre-trained aligners.

Requiring only minor task-specific adaptations, we apply this framework to three separate broad-coverage semantic parsing tasks: AMR, SDP, and UCCA. Experimental results show that our neural transducer outperforms the state-of-the-art parsers on AMR (77.0% F1 on LDC2017T10 and 71.3% F1 on LDC2014T12) and UCCA (76.6% F1 on the English-Wiki dataset v1.2), and is competitive with the state of the art on SDP (92.2% F1 on the English DELPH-IN MRS dataset).

6.2 Unified Transduction Problem

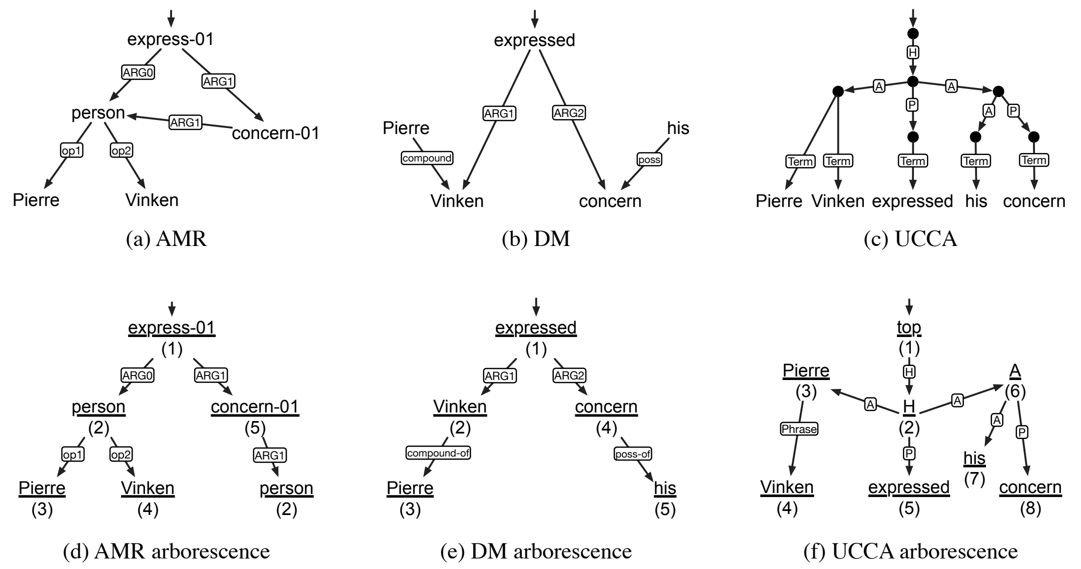


Figure 6.1: Meaning representation in the task-specific format – (a) AMR, (b) DM, and (c) UCCA – for an example sentence *"Pierre Vinken expressed his concern"*. Meaning representation (d), (e) and (f) are in the unified arborescence format, which are converted from (a), (b) and (c) respectively.

6.2.1 Unified Arborescence Format

We first introduce a unified target format for different broad-coverage semantic parsing tasks. Meaning representation in the unified format is an arborescence (aka, a directed rooted tree), which is converted from its corresponding task-specific semantic graph via the following *reversible* steps:

AMR Reentrancy is what can make an AMR graph not an arborescence (it introduces cycles). Following Chapter 5, we convert an AMR graph into an arborescence by duplicating nodes that have reentrant relations; that is, whenever a node has a reentrant relation, we make a copy of that node and use the copy to participate in the relation, thereby resulting in an arborescence. Next, in order to preserve the reentrancy information, we assign a node index to each node. Duplicated nodes are assigned the same index as the original node. Figure 6.1(d) shows an AMR arborescence converted from Figure 6.1(a): two "*person*" nodes have the same node index 2. The original AMR graph can be recovered by merging identically indexed nodes.

DM We first break the DM graph into a set of *weakly* connected subgraphs. For each subgraph, if it has the *top* node, we treat *top* as root; otherwise, we treat the node with the max outdegree as root. We then run depth-first traversal over each subgraph from its root to yield an arborescence, and repeat the following three steps until no more edges can be added to the arborescence: (1) we run breadth-first traversal over the arborescence from the root until we find a node that has an incoming edge not belonging to the arborescence; (2) we reverse the edge and add a `-of` suffix to the edge label; (3) we run depth-first search from that node to include more edges to the arborescence.

During the whole process, we add node indices and duplicate reentrant nodes in the same way as AMR conversion. Finally, we connect arborescences by adding a `null` edge from *top* to other arborescence roots. Figure 6.1(e) shows a DM arborescence converted from Figure 6.1(b). The original DM graph can be recovered by removing `null` edges, merging identically indexed nodes, and reversing edges with `-of` suffix.

UCCA To date, official UCCA evaluation only considers UCCA's *foundational* layer, which is already an arborescence. We convert it to the unified arborescence format by first collapsing subgraphs of pre-terminal nodes: we replace each pre-terminal node with its first terminal node; if the pre-terminal node has other terminals, we add a special `phrase` edge from the first terminal node to other terminal nodes. The collapsing step largely reduces the number of terminal nodes in UCCA. We then add labels to the remaining non-terminal nodes. Each node label is simply the same as its incoming edge label. We find that adding node labels improves performance of our neural transducer (See § 6.5.2 for the experimental results). Lastly, we add node indices in the same way as AMR conversion. Figure 6.1(f) shows a DM arborescence converted from Figure 6.1(c). The original UCCA DAG can be recovered by expanding pre-terminal subgraphs, and removing non-terminal node labels.

6.2.2 Problem Formalization

For any broad-coverage semantic parsing task, we denote the input text by X, and the output meaning representation in the unified arborescence format by Y, where X is a sequence of tokens $\langle x_1, x_2, ..., x_n \rangle$ and Y can be decomposed as a sequence of semantic relations $\langle y_1, y_2, ..., y_m \rangle$. A relation y is a tuple $\langle u, d^u, r, v, d^v \rangle$, consisting of a source node label u, a source node index d^u, a relation type r, a target node label v, and a target node index d^v.

Let $\mathcal{Y}$ be the *output space*. The unified transduction problem is to seek the most-likely sequence of semantic relations $\hat{Y}$ given X:

$$\begin{aligned}\hat{Y} &= \operatorname*{argmax}_{Y \in \mathcal{Y}} \mathrm{P}(Y \mid X) \\ &= \operatorname*{argmax}_{Y \in \mathcal{Y}} \prod_{i}^{m} \mathrm{P}(y_i \mid y_{<i}, X)\end{aligned}$$

6.3 Transducer

To tackle the unified transduction problem, we introduce an attention-based neural transducer that extends the sequence-to-graph (STOG) transducer in Chapter 5. STOG addresses semantic parsing in a two-stage process: it first employs an extended variant of pointer-generator network (See et al., 2017) to convert the input text into a list of nodes, and then uses a deep biaffine graph-based parser (Dozat and Manning, 2016) with a maximum spanning tree (MST) algorithm to create edges. In contrast, our attention-based neural transducer directly transduces the input text into a meaning representation in *one* stage via a sequence of semantic relations. A high-level model architecture of our transducer is depicted in Figure 6.2: an *encoder* first encodes the input text into hidden states; and then conditioned on the hidden states, at each decoding time step, a *decoder* takes the previous semantic relation as input, and outputs a new semantic relation, which includes a target node, a relation type, and a source node.

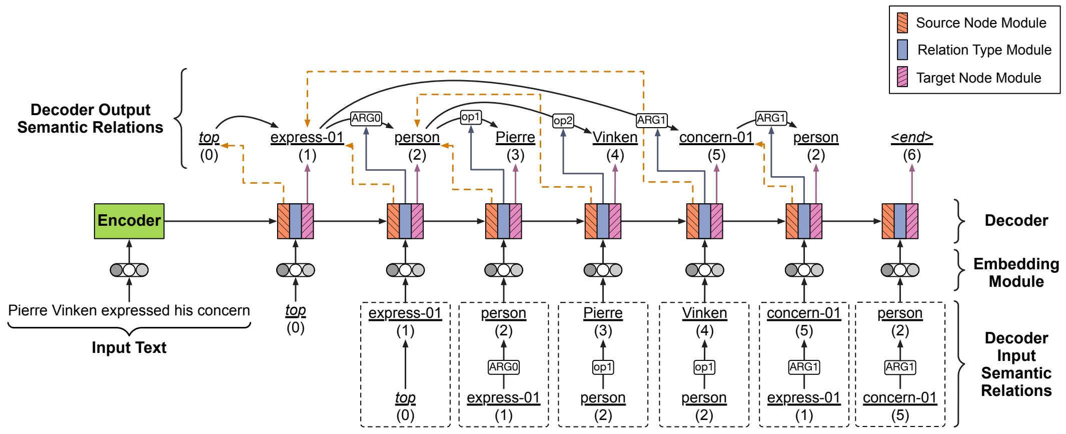


Figure 6.2: The encoder-decoder architecture of our attention-based neural transducer. An encoder encodes the input text into hidden states. A decoder is composed by three modules: a target node module, a relation type module, and a source node module. At each decoding time step, the decoder takes the previous semantic relation as input, and outputs a new semantic relation in a *factorized* way: firstly, the target node module produces a new target node; secondly, the source node module *points* to a preceding node as a new source node; finally, the relation type module predicts the relation type between source and target nodes.

There is a significant difference between STOG and our model: STOG first predicts nodes, and then edges. These two stages are done *separately* (except that a shared encoder is used). At the node prediction stage, their model has no knowledge of edges, and therefore node prediction is performed purely based previous nodes. At the edge prediction stage, their model predicts the head of each node in parallel. Head prediction of one node has no constrains or impact on another. As a result, MST algorithms have to be used to search for a valid prediction. In comparison, our model does not have two separate stages for node and edge prediction. At each decoding step, our model predicts not only a node, but also the incoming edge to the node, which includes a source and a relation type. See Figure 6.2 for an example. The predicted node and incoming edge together with previous predictions form a partial semantic graph, which is used as input of the next decoding step for the next node and incoming edge prediction. Our model therefore makes predictions based on the partial semantic graph, which helps prune the output space for both nodes and edges. Since at each decoding step, we assume the incoming edge is always from a preceding node (see § 6.3.3 for the details), the predicted semantic graph is guaranteed to be a valid arborescence, and a MST algorithm is no longer needed.

6.3.1 Encoder

At the encoding stage, we employ an encoder embedding module to convert the input text into vector representations, and a BiLSTM is used to encode vector representations into hidden states.

Encoder Embedding Module concatenates word-level embeddings. They come from GloVe (Pennington et al., 2014) and BERT[2] (Devlin et al., 2018), char-level embeddings from CharCNN (Kim et al., 2016), and randomly initialized embeddings for POS tags.

For AMR, it includes extra embeddings for anonymization indicators that tell the encoder whether a token is an anonymized token from preprocessing.

For UCCA, it includes extra randomly initialized embeddings for NER tags, syntactic dependency labels, punctuation indicators, and shapes that are provided in the UCCA official dataset.

[2]We use average pooling in the same way as STOG to get word-level embeddings from BERT.

Multi-layer BiLSTM (Hochreiter and Schmidhuber, 1997) is defined as:

$$\mathbf{s}_t^l = \begin{bmatrix} \overrightarrow{\mathbf{s}}_t^l \\ \overleftarrow{\mathbf{s}}_t^l \end{bmatrix} = \begin{bmatrix} \overrightarrow{\text{LSTM}}(\mathbf{s}_t^{l-1}, \mathbf{s}_{t-1}^l) \\ \overleftarrow{\text{LSTM}}(\mathbf{s}_t^{l-1}, \mathbf{s}_{t+1}^l) \end{bmatrix}, \tag{6.1}$$

where $\mathbf{s}_t^l$ is the l-th layer hidden state at time step t; $\mathbf{s}_i^t$ is the embedding module output for token x_t.

6.3.2 Decoder

Decoder Embedding Module at decoding time step i converts elements in the input semantic relation $\langle u_i, d_i^u, r_i, v_i, d_i^v \rangle$ into vector representations $\langle \mathbf{u}_i, \mathbf{d}_i^u, \mathbf{r}_i, \mathbf{v}_i, \mathbf{d}_i^v \rangle$:[3]

$\mathbf{u}_i$ and $\mathbf{v}_i$ are concatenations of word-level embeddings from GloVe, char-level embeddings from CharCNN, and randomly initialized embeddings for POS tags. POS tags for source and target nodes are inferred at runtime: if a node is copied from input text, the POS tag of the corresponding token is used; if it is copied from a preceding node, the POS tag of the preceding node is used; otherwise, an `UNK` tag is used.

$\mathbf{d}_i^u$, $\mathbf{d}_i^v$ and $\mathbf{r}_i$ are randomly initialized embeddings for source node index, target node index, and relation type.

Next, the decoder outputs a new semantic relation in a *factorized* way depicted in Figure 6.2: First, a target node module takes vector representations of the previous semantic relation, and predicts a target node label as well as its index. Then, a source node module predicts a source node via *pointing* to a

[3]While training, the input semantic relation is from the reference sequence of relations; at test time, it is the previous decoder output semantic relation.

preceding node. Lastly, a relation type module takes the predicted source and target nodes, and predicts the relation type between them.

Target Node Module converts vector representations of the input semantic relation into a hidden state $\mathbf{z}_i$ in the following way:

$$\mathbf{z}_i = \text{FFN}^{(\text{relation})}([\mathbf{h}_i^l; \mathbf{c}_i; \mathbf{r}_i; \mathbf{u}_i; \mathbf{d}_i^u]) \tag{6.2}$$

$$\mathbf{h}_i^l = \text{LSTM}(\mathbf{h}_i^{l-1}, \mathbf{h}_{i-1}^l) \tag{6.3}$$

$$\text{FFN}(\mathbf{x}) = \mathbf{W}\mathbf{x} + \mathbf{b} \tag{6.4}$$

where an l-layer LSTM generates contextual representation $\mathbf{h}_i^l$ for *target* node v_i (for initialization, $\mathbf{h}_i^0 = [\mathbf{v}_i; \mathbf{d}_i^v]$, $\mathbf{h}_0^l = [\overleftarrow{\mathbf{s}}_1^l; \overrightarrow{\mathbf{s}}_n^l]$). A feed-forward neural network $\text{FFN}^{(\text{relation})}$ generates the hidden state $\mathbf{z}_i$ of the input semantic relation by combining contextual representation $\mathbf{h}_i^l$ for target node v_i, encoder context vector $\mathbf{c}_i$, and vector representations $\mathbf{r}_i, \mathbf{u}_i, \mathbf{d}_i^u$ for relation type r_i, source node label u_i and source node index d_i^u.

Encoder context vector $\mathbf{c}_i$ is a weighted-sum of encoder hidden states $\mathbf{s}_{1:n}^l$. The weight is attention $\mathbf{a}_i^{(\text{enc})}$ from the decoder at decoding step i to encoder hidden states:

$$\mathbf{a}_i^{(\text{enc})} = \text{softmax}\big(\text{MLP}^{(\text{enc})}([\mathbf{h}_i^l; \mathbf{s}_{1:n}^l])\big) \tag{6.5}$$

$$\text{MLP}(\mathbf{x}) = \text{ELU}(\mathbf{W}\mathbf{x} + \mathbf{b}) \tag{6.6}$$

Given the hidden state $\mathbf{z}_i$ for input semantic relation, we use an extended variant of pointer-generator network to compute the probability distribution of next target node label v_{i+1}:

$$\mathrm{P}(v_{i+1}) = p_{\text{gen}}\mathbf{p}_i^{(\text{vocab})} \oplus p_{\text{enc}}\mathbf{a}_i^{(\text{enc})} \oplus p_{\text{dec}}\mathbf{a}_i^{(\text{dec})} \tag{6.7}$$

$$\mathbf{p}_i^{(\text{vocab})} = \text{softmax}\big(\text{FFN}^{(\text{vocab})}(\mathbf{z}_i)\big) \tag{6.8}$$

$$\mathbf{a}_i^{(\text{dec})} = \text{softmax}\big(\text{MLP}^{(\text{dec})}([\mathbf{z}_i; \mathbf{z}_{1:i-1}])\big) \tag{6.9}$$

$$[p_{\text{gen}}, p_{\text{enc}}, p_{\text{dec}}] = \text{softmax}\big(\text{FFN}^{(\text{switch})}(\mathbf{z}_i)\big) \tag{6.10}$$

$\mathrm{P}(v_{i+1})$ is a hybrid of three parts: (1) emitting a new node label from a pre-defined vocabulary via probability distribution $\mathbf{p}_i^{(\text{vocab})}$; (2) copying a token from the encoder input text as node label via encoder-side attention $\mathbf{a}_i^{(\text{enc})}$; and (3) copying a node label from preceding target nodes via decoder-side attention $\mathbf{a}_i^{(\text{dec})}$. Scalars $p_{\text{gen}}, p_{\text{enc}}$ and p_{dec} act as a soft switch to control the production of target node label from different sources.

The next target node index d^v_{i+1} is assigned based on the following rule:

$$d^v_{i+1} = \begin{cases} d^v_j, \text{ if } v_{i+1} \text{ copies its antecedent } v_j. \\ i+1, \text{ otherwise.} \end{cases}$$

Source Node Module produces the next source node label u_{i+1} via *pointing* to a node label among preceding *target* node labels (the dotted arrows shown in Figure 6.2). The probability distribution of next source node label u_{i+1} is defined as

$$\mathrm{P}(u_{i+1}) = \text{softmax}\big(\text{BIAFFINE}(\mathbf{h}_{i+1}^{(\text{start})}, \mathbf{h}_{1:i}^{(\text{end})})\big) \tag{6.11}$$

where BIAFFINE is a biaffine function (Dozat and Manning, 2016). $\mathbf{h}_{i+1}^{(\text{start})}$ is the vector representation for the *start* of the pointer. $\mathbf{h}_{1:i}^{(\text{end})}$ are vector representations for possible *ends* of the pointer. They are computed by two multi-layer perceptrons:

$$\mathbf{h}_{i+1}^{(\text{start})} = \text{MLP}^{(\text{start})}(\mathbf{h}_{i+1}^{l}) \tag{6.12}$$

$$\mathbf{h}_{1:i}^{(\text{end})} = \text{MLP}^{(\text{end})}(\mathbf{h}_{1:i}^{l}) \tag{6.13}$$

Note that $\mathbf{h}_{i+1}^{l}$ is the LSTM hidden state for target node v_{i+1}, generated by Equation (6.3) in the target node module. We reuse LSTM hidden states from the target node module such that we can train the decoder modules jointly.

Then, the next source node index d_{i+1}^{u} is the same as the target node the module points to.

Relation Type Module also reuses LSTM hidden states from the target node module to compute the probability distribution of next relation type r_{i+1}. Assuming that the source node module points to target node label v_j as the next source node label, The next relation type probability distribution is computed by:

$$\text{P}(r_{i+1}) = \text{softmax}\big(\text{BILINEAR}(\mathbf{h}_{i+1}^{(\text{rel-src})}, \mathbf{h}_{i+1}^{(\text{rel-tgt})})\big) \tag{6.14}$$

$$\mathbf{h}_{i+1}^{(\text{rel-src})} = \text{MLP}^{(\text{rel-src})}(\mathbf{h}_{j}^{l}) \tag{6.15}$$

$$\mathbf{h}_{i+1}^{(\text{rel-tgt})} = \text{MLP}^{(\text{rel-tgt})}(\mathbf{h}_{i+1}^{l}) \tag{6.16}$$

6.3.3 Training

To ensure that at each decoding step, the source node can be found in the preceding nodes, we create the reference sequence of semantic relations by running a *pre-order* traversal over the reference arborescence. The pre-order traversal only determines the order between a node and its children. As for the order of its children, we sort them in alphanumerical order in the case of AMR, following Chapter 5. In the case of SDP, we sort the children based on their order in the input text. In the case of UCCA, we sort the children based on their UCCA node ID.

Given a training pair $\langle X, Y\rangle$, the optimization objective is to maximize the decomposed conditional log likelihood $\sum_i \log\big(\mathrm{P}(y_i \mid y_{<i}, X)\big)$, which is approximated by:

$$\sum_i \log\big(\mathrm{P}(u_i)\big) + \log\big(\mathrm{P}(r_i)\big) + \log\big(\mathrm{P}(v_i)\big) \tag{6.17}$$

We also employ label smoothing (Szegedy et al., 2016) to prevent overfitting, and include a coverage loss (See et al., 2017) to penalize repetitive nodes: $\mathrm{covloss}_i = \sum_t \min(\mathbf{a}_i^{(\mathrm{enc})}[t], \mathbf{cov}_i[t])$, where $\mathbf{cov}^i = \sum_{j=0}^{i-1} \mathbf{a}_j^{(\mathrm{enc})}$.

6.3.4 Prediction

Our transducer at each decoding time step looks for the source node from the preceding nodes, which ensures that the output of a greedy search is already a valid arborescence $\hat{Y}$:

$$\mathrm{P}(\hat{Y} \mid X) = \prod_i \max_{u_i} \mathrm{P}(u_i) \max_{r_i} \mathrm{P}(r_i) \max_{v_i} \mathrm{P}(v_i)$$

Algorithm 3: Beam Search over Semantic Relations.

```
Input    : The input text $X$.
Output: A sequence of relations $Y = \{y_1, ...y_m\}$.
// Initialization.
$i$, score $\leftarrow 0, 0$;
$Y$, finished $\leftarrow \{\}, \{\}$;
beam $\leftarrow \{\{Y, \text{score}\}\}$;
// Encoding.
encode($X$);
// Decoding.
for $i \leftarrow 1$ to MaxLength do
    new_beam $\leftarrow \{\}$;
    $\{Y, \text{score}\}$ = beam.pop();
    for $v_i$ in topK($\text{P}(v_i)$) do
        if $v_i$ = EOS then
            finished.push($\{Y, \text{score}\}$);
        else
            for $u_i \leftarrow v_0$ to $v_{i-1}$ do
                for $r_i$ in RelationTypeSet do
                    $Y \leftarrow Y \cup \{\langle u_i, r_i, v_i \rangle\}$;
                    score $\leftarrow$ score $+ \log\big(\text{P}(u_i)\big) + \log\big(\text{P}(r_i)\big) + \log\big(\text{P}(v_i)\big)$;
                    new_beam.push({Y, score});
                end
            end
        end
    end
    beam $\leftarrow$ new_beam.topK();
end
// Finishing.
while beam.not_empty() do
    $\{Y, \text{score}\} \leftarrow$ beam.pop();
    finished.push($\{Y, \text{score}\}$);
end
$\{Y, \text{score}\} \leftarrow$ finished.topK(k=1);
return $Y$;
```

Therefore, a MST algorithm such as the Chu-Liu-Edmonds algorithm at $\mathcal{O}(EV)$ used in Chapter 5 is no longer needed, and the decoding speed of our transducer is $\mathcal{O}(V)$. E denotes the number of edges. V the number of nodes. Moreover, since our transducer builds the meaning representation via a sequence of semantic relations, we implement a beam search over relation in Algo. 3. Compared to the beam search used in Chapter 5 that only returns top-k nodes, our beam search finds the top-k relation scores, which includes source nodes, relation types and target nodes.

6.4 Data Pre- and Post-processing

AMR Pre- and post-processing steps are similar to those of Chapter 5: in preprocessing, we anonymize subgraphs of entities, remove senses, and convert resultant AMR graphs into the unified format; in post-processing, we assign the most frequent sense for nodes, restore Wikipedia links using the DBpedia Spotlight API (Daiber et al., 2013), add polarity attributes based on rules observed from training data, and recover the original AMR format from the unified format.

DM No pre- or post-processing is done to DM except converting them into the unified format, and recovering them from predictions.

UCCA During training, multi-sentence input text and its corresponding DAG are split into single-sentence training pairs based on rules observed from training data. At test time, we split multi-sentence input text, and join the predicted graphs into one. We also convert the original format to the unified format in preprocessing, and recover the original DAG format in post-processing.

Hidden Size		
Glove	300	
BERT	1024	
POS / NER / Dep / Shapes	100	
Anonymization / Node index	50	
CharCNN kernel size	3	
CharCNN channel size	100	
Encoder	2@512	
Decoder	2@1024	
Biaffine input size	256	
Bilinear input size	AMR	128
	DM	256
	UCCA	128
Optimizer		
Type	ADAM	
Learning rate	0.001	
Maximum gradient norm	5.0	
Coverage loss weight λ	1.0	
Label smoothing ϵ	0.1	
Beam size	5	
Batch size	64	
Dropout rate	AMR	0.33
	DM	0.2
	UCCA	0.33
Vocabulary		
Encoder-side vocab size	AMR 1.0	9200
	AMR 2.0	18000
	DM	11000
	UCCA	10000
Decoder-side vocab size	AMR 1.0	7300
	AMR 2.0	12200
	DM	11000
	UCCA	10000

Table 6.1: Hyperparameter settings

6.5 Experiments

6.5.1 Data and Setup

We evaluate our approach on three separate broad-coverage semantic parsing tasks: (1) AMR 2.0 (LDC2017T10) and 1.0 (LDC2014T12); (2) the English DM dataset from SemEval 2015 Task 18 (LDC2016T10); (3) the UCCA English Wikipedia Corpus v1.2 (Abend and Rappoport, 2013; Hershcovich et al., 2019). The train/dev/test split follows the official setup. Our model is trained on two GeForce GTX TITAN X GPUs with early stop based on the dev set. We fix BERT parameters due to the limited GPU memory. Hyperparameter setting for each task is provided in Table 6.1.

6.5.2 Results

AMR Table 6.2 compares our neural transducer to the previous best results (SMATCH F1, Cai and Knight, 2013) on AMR test sets. The transducer improves the state of the art on AMR 2.0 by 0.7% F1. On AMR 1.0 where training data is much smaller than AMR 2.0, it shows a larger improvement (1.1% F1) over the state of the art.

In Table 6.2, we also conduct ablation study on beam search to investigate contributions from the model architecture itself and the beam search algorithm. The transducer model without beam search is already better than the previous best parser that is equipped with beam search. When compared with the previous best parser without beam search, our model still has around 1.0% F1 improvement.

Data	Parser	F1(%)
AMR 2.0	Cai and Lam (2019)	73.2
	Lyu and Titov (2018)	74.4±0.2
	Lindemann et al. (2019)	75.3±0.1
	Naseem et al. (2019)	75.5
	STOG*	76.3±0.1
	- w/o beam search	75.3±0.1
	Ours	**77.0**±0.1
	- w/o beam search	76.4±0.1
AMR 1.0	Flanigan et al. (2016)	66.0
	Pust et al. (2015)	67.1
	Wang and Xue (2017)	68.1
	Guo and Lu (2018)	68.3±0.4
	STOG*	70.2±0.1
	- w/o beam search	69.2±0.1
	Ours	**71.3**±0.1
	- w/o beam search	70.4±0.1

Table 6.2: SMATCH F1 on AMR 2.0 and 1.0 test sets. Standard deviation is computed over 3 runs. *STOG refers to the sequence-to-graph transduction approach we propose in Chapter 5.

Metric	L'18	N'19	STOG	Ours
SMATCH	74	75	76	**77**
Unlabeled	77	**80**	79	**80**
No WSD	76	76	77	**78**
Reentrancies	52	56	60	**61**
Concepts	**86**	**86**	85	**86**
Named Ent.	**86**	83	78	79
Wikification	76	80	**86**	**86**
Negation	58	67	75	**77**
SRL	70	**72**	70	71

Table 6.3: Fine-grained F1 scores on the AMR 2.0 test set. L'18 is Lyu and Titov (2018); N'19 is Naseem et al. (2019); STOG is the sequence-to-graph transduction approach proposed in Chapter 5.

Table 6.3 summarizes the parser performance on each subtask using Damonte et al. (2017) evaluation tool. Our transducer outperforms STOG on all subtasks, but is still not close to Lyu and Titov (2018) on named entities due to the different preprocessing methods for anonymization.

Parser	ID	OOD
Du et al. (2015)	89.1	81.8
Almeida and Martins (2015)$^{(\text{open})}$	89.4	83.8
Wang et al. (2018)	90.3	84.9
Peng et al. (2017a): BASIC	89.4	84.5
Peng et al. (2017a): FREDA3	90.4	85.3
Peng et al. (2018)	91.2	86.6
Dozat and Manning (2018)	**93.7**	**88.9**
Ours	92.2	87.1

Table 6.4: Labeled F1 (%) scores on the English DM in-domain (WSJ) and out-of-domain (Brown corpus) test sets. $^{(\text{open})}$ denotes results from the open track.

DM Table 6.4 compares our neural transducer to the state of the art (labeled F1) on the English DM in-domain (ID) and out-of-domain (OOD) data. Except Dozat and Manning (2018), our transducer outperforms all other baselines, including FREDA3 of Peng et al. (2017a) and Peng et al. (2018), which leverage multi-task learning from different datasets. The best parser (Dozat and Manning, 2018) is specifically designed for bi-lexical dependencies, and is not directly applicable to other semantic parsing tasks such as AMR and UCCA. In contrast, our transducer is more general, and is competitive to the best SDP parser.

UCCA Table 6.5 compares our results to the previous best published results (labeled F1 for all edges) on the English Wiki test set. The best performance is from Jiang et al. (2019), where they convert UCCA graphs to constituency trees,

Parser	F1 (%)
Hershcovich et al. (2017)	71.1
Hershcovich et al. (2018): single	71.2
Hershcovich et al. (2018): MTL	74.3
Jiang et al. (2019)	**80.5**
Ours	76.6±0.1
- w/o non-terminal node labels	75.7±0.1

Table 6.5: Labeled F1 (%) scores for all edges including primary edges and remote edges. Standard deviation is computed over 3 runs.

and train a framework for constituency parsing and remote edge recovery. Hershcovich et al. (2018) explore multi-task learning (MTL) to improve UCCA parsing, using AMR, DM and UD parsing as auxiliaries. While improvement is achieved UCCA parsing, their MTL model shows poor results on the auxiliary tasks: 64.7% unlabeled F1 on AMR, 27.2% unlabeled F1 on DM, and 4.9% UAS on UD. In comparison, our transducer improves the state of the art on AMR, and shows competitive results on DM. At the same time, it also shows reasonable results on UCCA. When converting UCCA DAGs to the unified format, we adopt a simple rule (§ 6.2.1) to add node labels to non-terminals. Table 6.5 shows that these node labels do improve the parsing performance from 75.7% to 76.6%.

6.5.3 Analysis

Validity Graph-based parsers like Dozat and Manning (2018); Zhang et al. (2019a) make independent decisions on edge types. As a result, the same outgoing edge type can appear multiple times to a node. For instance, a node can have more than one `ARG1` outgoing edge. Although F1 scores can be computed for graphs with such kind of nodes, these graphs are in fact invalid mean representations. Our neural transducer incrementally builds meaning representations: at each decoding step, it takes a semantic relation as input, and has memory of preceding edge type information, which implicitly places constraints on edge type prediction. We compute the number of invalid graphs predicted by STOG and our neural transducer on the AMR 2.0 test set, and find that our neural transducer reduces the number of invalid graphs by 8%.

Speed Besides the improvement on parsing accuracy, we also significantly speed up parsing. Table 6.6 compares the parsing speed of our transducer and STOG on the AMR 2.0 test set, under the same environment setup. Without relying on MST algorithms to produce a valid arborescence, our transducer is able to parse at 1.7x speed.

	Speed (tokens/sec)
STOG	617
Ours	**1076**

Table 6.6: Parsing speed on the AMR 2.0 test set.

6.6 Conclusion

We cast three broad-coverage semantic parsing tasks into a unified transduction framework, and propose a neural transducer to tackle the problem. Given the input text, the transducer incrementally builds a meaning representation via a sequence of semantic relations. Experiments conducted on three tasks show that our approach improves the state of the art in both AMR and UCCA, and is competitive to the best parser in SDP. Compared with transition-based parsers (e.g. Damonte et al., 2017) and graph-based parsers (e.g. Dozat and Manning, 2018), our transductive framework does not require a pre-trained aligner, and it is capable of building a meaning representation that is less anchored to the input text. This work can be viewed as a starting point for cross-framework semantic parsing.

Chapter 7

Conclusions

In this thesis, we present a series of transductive parsing architectures for two semantic parsing problems: cross-lingual semantic parsing (Part I) and broad-coverage semantic parsing (Part II).

In Chapter 1 and 2, we walk through the history of semantic parsing from hand-coding rules to deep learning methods, and describe the renewed interest in designing general-purpose meaning representations in response to previous efforts on semantic parsing that are limited to specific domains. We provide summary background of several recent general-purpose meaning representation frameworks: Abstract Meaning Representation (AMR; Banarescu et al., 2013), Universal Conceptual Cognitive Annotation (UCCA; Abend and Rappoport, 2013), Semantic Dependency Parsing (SDP; Oepen et al., 2014; Oepen et al., 2015), and Universal Decompositional Semantics (UDS; White et al., 2016). We focus on several new challenges in developing semantic parsing systems for the general-purpose meaning representations: lexical mismatch, structural complexity, framework balkanization, and limited data. We then review the related work on parsing for each meaning representation framework.

Next, we introduce a pattern-based predicate-argument extraction tool – PredPatt – which automatically provides shallow predicate-argument semantics from raw sentences.

In Part I, we are concerned with representing semantics of multiple natural languages in a single semantic analysis.

In Chapter 3, we focus on shallow semantics; that is, predicate-argument structures, or open information extraction (Open IE). We introduce the task of cross-lingual Open IE: distilling facts from foreign language into shallow semantic representations in another language. To tackle this task, we first propose a joint approach based on a neural sequence-to-sequence model. Then we improve our joint approach via a novel selective decoding mechanism and a simple and effective learning method *Halo*. In the experiments, we compare our joint approaches with the traditional pipeline that first translates foreign text and then runs monolingual Open IE tools. We show that our approaches achieve consistent and significant improvements over the pipeline in a variety of cross-lingual open IE scenarios.

In Chapter 4, we move to a form of universal decompositional semantic (UDS) analysis, which is designed to allow systems to target varying levels of structural complexity. We introduce the task of cross-lingual decompositional semantic parsing. In this new task, we present three forms of UDS analysis, graph, linearized and flat forms, which are created for different purposes and inter-convertible; We design a new evaluation metric that is better at differentiating two UDS graph representations; We propose an end-to-end learning approach with a novel annotating mechanism that supports intra-sentential coreference; We create a Chinese-English decompositional semantic parsing dataset. Our end-to-end approach outperforms strong baselines on the new evaluation metric. We separately evaluate the coreference mechanism, Semantic Proto-Role and event factuality prediction, showing promising results as well.

In Part II, we develop efficient transductive architectures for broad-coverage semantic parsing whose target representations have rich sentence-level semantics.

In Chapter 5, we focus on Abstract Meaning Representation (AMR) parsing. We first summarize three major challenges in AMR parsing: (1) the property of reentrancy, (2) the lack of alignments between nodes and words, and (3) relattively limited amounts of labeled data. We then propose an attention-based model that treats AMR parsing as a two-stage sequence-to-graph transduction problem. By leveraging attention mechanisms and pre-trained language models, the proposed parser is aligner-free, and it can be effectively trained with limited amounts of AMR training data. Experiments on publicly available AMR corpora show that our parser outperforms all existing AMR parsers by a significant margin. We also do the ablation study and analysis, showing the effectiveness of each novel component we introduce in the parser.

In Chapter 6, we extend our transductive parsing framework to cover other broad-coverage semantic parsing tasks. We introduce a unified target format – Unified Arborescence Format – for different general-purpose meaning representations. Based on this unified format, we propose a refined a transduction paradigm, which directly transduces natural language text into a meaning representation in a *single* stage. The core of this approach is an attention-based neural model, which incrementally builds a meaning representation via a sequence of semantic relations. We conduct experiments on three separate broad-coverage semantic parsing tasks – AMR, SDP and UCCA. Experiment results show that our parser improves the state of the art on both AMR and UCCA, and is competitive with the state of the art on SDP. Analysis reveals that our parser has higher validity and speed.

The efforts on transductive semantic parsing presented in the thesis pave the way for the following future research directions:

Better Transductive Semantic Parsing Our transductive semantic parsing model benefits from the pre-trained Transformer model like BERT (Devlin et al., 2018). Specifically, in Chapter 5, BERT embeddings are concatenated with other type of embeddings to form the vector representation for input words. However, our transductive model and BERT have different granularities in terms of input: our model takes words as input, but BERT takes subword units as input. This means that one word input to our model may corresponds to multiple embeddings of BERT. In order to accurately use BERT embeddings to represent word-level input, our solution in Chapter 5 is to apply an average pooling function to the output of BERT (as illustrated in Figure 7.1). While the average pooling solution provides a workaround for the granularity difference issue and improves experiment results in § 5.7, it has two obvious limits: (1) the extra pooling layer introduces extra computation and parameters[1] that slows down both training and inference time; (2) no matter what pooling function

[1]In order to perform average pooling in batch, masks are used to keep track of the start and end positions of each average pooling operation.

is used, the information loss is inevitable and the resulting embeddings will have less information than the orginial BERT subword embeddings.

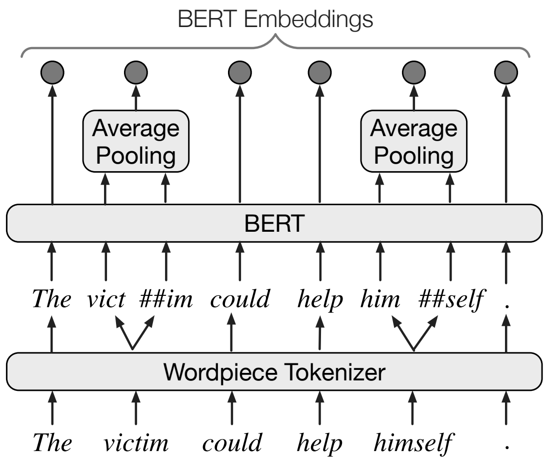

BERT Embeddings
Average Pooling
Average Pooling
BERT
The vict ##im could help him ##self .
Wordpiece Tokenizer
The victim could help himself .

Figure 7.1: Word-level embeddings generated by average pooling over BERT.

A better solution to explore in the future is to change the input granularity of our transductive model to subword units. To do so, two important issues need to be resolved: (1) How to use word-level features in subword units? We introduce features such as POS (part-of-speech) tags and anonymization tags in the input, but they are all annotated at word level. In order to use them in subword units, a simple solution to assign each unit the same tag as the word it belongs to. (2) How to perform source- and target-side copy mechanisms over subword units? These copy mechanisms has shown to be critical to node generation, and they are all designed to work at word level. Breaking input words into subword units may substantially reduce the number of nodes that can be copied, and thus limits the use of source- and target-side copy mechanisms. A solution to adapting copy mechanisms to subword units is to introduce an end-of-word indicator for each subword. An end-of-word indicator is binary, indicating whether the corresponding subword is the end of a word. When these copy mechanisms compute attention over subword units, these end-of-word indicators can be used to mask out attention on subword units that are not the end of a word. At the same time, we maintain a look-up table from end-of-word subword units (as well as position information) to their corresponding words. When a end-of-word subword gets the highest attention weight, the copy mechanism simply copies the corresponding word from the look-up table. This solution does not introduction extra computation or parameters at the training time (assuming we still use maximum likelihood estimation as in Chapter 5 and 6). The look-up table is only necessary at the inference time.

Once our transductive parsing model is changed to take subword units as input, the LSTM encoder and decoder can be replaced with Transformer layers, which enables a tight integration with the pre-trained Transformer model and saves a large amount of parameters.

Joint Learning Previous efforts on joint learning of syntactic and semantic representations have shown that one learned task is helpful in improving the other (Lluís and Màrquez, 2008; Lluís et al., 2013; Henderson et al., 2013; Swayamdipta et al., 2016; Swayamdipta et al., 2018). In our transductive parsing framework, the encoder-decoder architecture is suitable for jointly learning both syntactic and semantic parsing. On one hand, graph-based approaches that leverage scoring functions such as BIAFFINE over a single encoder have shown the state-of-the-art performance on *bi-lexical* dependency parsing (Dozat and Manning, 2016) and semantic parsing whose target representations are directly anchored to the input text (Dozat and Manning, 2018;

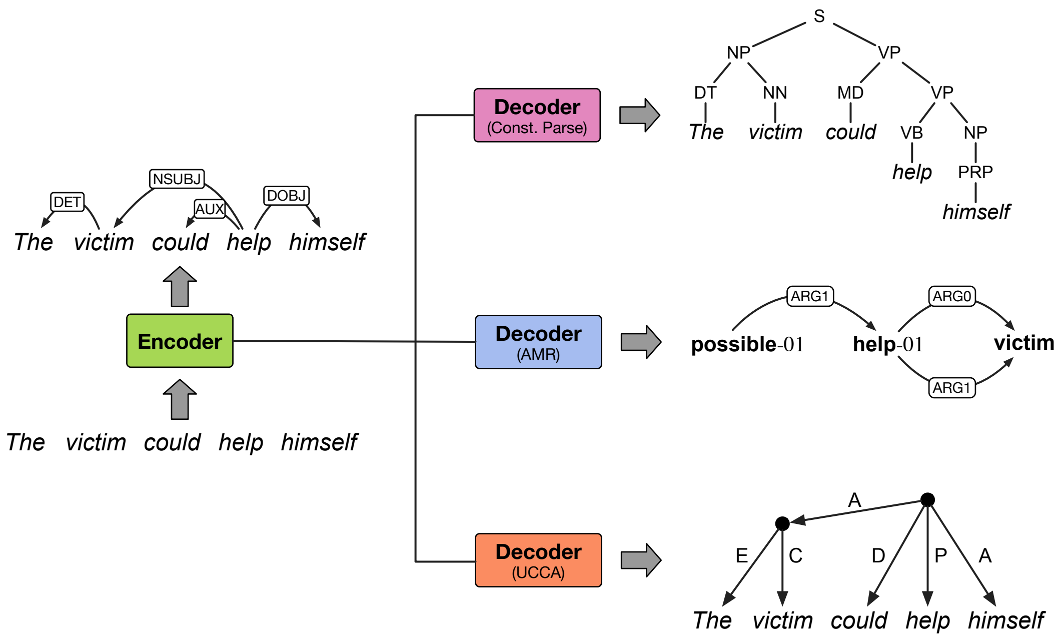


Figure 7.2: Jointly learning the transductive parsing framework.

Wang et al., 2019). On the other hand, we have shown in the thesis that varied copy mechanisms used by the decoder are very effective at reducing data sparsity and resolving reentrancies in semantic parsing whose target representations abstract away from input text. Therefore, it is feasible to jointly learn syntactic and semantic parsing tasks under our proposed encoder-decoder framework.

Figure 7.2 illustrates the high-level idea of the joint learning architecture: on the source side, the encoder encodes input text and predicts bi-lexical (semantic) dependency relations over input words, the learning objective of which is denoted by $\mathcal{L}_{\text{bilex}}$; on the target side, different decoders are employed to perform structural predictions for different tasks. In Chapter 6, we show how to compute the learning objectives for AMR parsing $\mathcal{L}_{\text{AMR}}$ and UCCA parsing $\mathcal{L}_{\text{UCCA}}$. Similarly, we introduce another decoder for constituency parsing $\mathcal{L}_{\text{const}}$, since constituency parses also have tree structures that can be easily converted into the unified arborescence format introduced in Chapter 6. Thus, the joint learning objective is denoted by:

$$\mathcal{L} = \mathcal{L}_{\text{bilex}} + \mathcal{L}_{\text{const}} + \mathcal{L}_{\text{AMR}} + \mathcal{L}_{\text{UCCA}}$$

At the same time, we appreciate the recent efforts (Oepen et al., 2019) on packaging distinct meaning representation frameworks into a uniform graph abstraction and serialization, which we can directly adopt to facilitate jointly learning of multiple semantic parsing tasks in our transductive framework.

Another direction to explore in joint learning is to use graph convolutional networks (GCNs; Defferrard et al., 2016; Kipf and Welling, 2016) on the source side of our transductive parsing framework. In the joint learning architecture illustrated in Figure 7.2, bi-lexical (semantic) dependency parsing tasks are performed solely on top of the encoder. No copy mechanisms are needed since dependency parses are directly anchored on the input words. Therefore, our model in practise is able to finish the dependency parsing tasks before decoding, which allows us to have more informative vector representations for input words. On one hand, our encoder generates contextual vector representations for input words based on their order in the input sentence. On the other hand, we can leverage GCNs to compute vector representations for input words based on their dependency graphs. Concatenation of these two types of vector representations provides vector representations that have both sentential and graphical context, and thus can be beneficial to the decoder-side structural prediction tasks as well as the copy mechanisms.

References

Abend, Omri and Ari Rappoport (2013). “Universal Conceptual Cognitive Annotation (UCCA)”. In: *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. Sofia, Bulgaria: Association for Computational Linguistics, pp. 228–238.

Abend, Omri and Ari Rappoport (2017). “The State of the Art in Semantic Representation”. In: *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. Vancouver, Canada: Association for Computational Linguistics, pp. 77–89.

Abzianidze, Lasha, Johannes Bjerva, Kilian Evang, Hessel Haagsma, Rik van Noord, Pierre Ludmann, Duc-Duy Nguyen, and Johan Bos (2017). “The Parallel Meaning Bank: Towards a Multilingual Corpus of Translations Annotated with Compositional Meaning Representations”. In: *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 2, Short Papers*. Valencia, Spain: Association for Computational Linguistics, pp. 242–247.

Almeida, Mariana S. C. and André F. T. Martins (2015). “Lisbon: Evaluating TurboSemanticParser on Multiple Languages and Out-of-Domain Data”. In: *Proceedings of the 9th International Workshop on Semantic Evaluation (SemEval 2015)*. Denver, Colorado: Association for Computational Linguistics, pp. 970–973.

Amodei, Dario, Sundaram Ananthanarayanan, Rishita Anubhai, Jingliang Bai, Eric Battenberg, Carl Case, Jared Casper, Bryan Catanzaro, Qiang Cheng, Guoliang Chen, et al. (2016). “Deep speech 2: End-to-end speech recognition in english and mandarin”. In: *International Conference on Machine Learning*, pp. 173–182.

Andor, Daniel, Chris Alberti, David Weiss, Aliaksei Severyn, Alessandro Presta, Kuzman Ganchev, Slav Petrov, and Michael Collins (2016). “Globally Normalized Transition-Based Neural Networks”. In: *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1:*

Long Papers). Berlin, Germany: Association for Computational Linguistics, pp. 2442–2452.

Angeli, Gabor, Melvin Jose Johnson Premkumar, and Christopher D. Manning (2015). "Leveraging Linguistic Structure For Open Domain Information Extraction". In: *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*. Beijing, China: Association for Computational Linguistics, pp. 344–354.

Arpit, Devansh, Stanislaw Jastrzkebski, Nicolas Ballas, David Krueger, Emmanuel Bengio, Maxinder S Kanwal, Tegan Maharaj, Asja Fischer, Aaron Courville, Yoshua Bengio, et al. (2017). "A Closer Look at Memorization in Deep Networks". In: *International Conference on Machine Learning*, pp. 233–242.

Artzi, Yoav, Kenton Lee, and Luke Zettlemoyer (2015). "Broad-coverage CCG Semantic Parsing with AMR". In: *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*. Lisbon, Portugal: Association for Computational Linguistics, pp. 1699–1710.

Artzi, Yoav and Luke Zettlemoyer (2013). "Weakly Supervised Learning of Semantic Parsers for Mapping Instructions to Actions". In: *Transactions of the Association for Computational Linguistics* 1, pp. 49–62.

Bahdanau, Dzmitry, Kyunghyun Cho, and Yoshua Bengio (2014). "Neural machine translation by jointly learning to align and translate". In: *arXiv preprint arXiv:1409.0473*.

Baldridge, Jason and Geert-Jan Kruijff (2002). "Coupling CCG and Hybrid Logic Dependency Semantics". In: *Proceedings of 40th Annual Meeting of the Association for Computational Linguistics*. Philadelphia, Pennsylvania, USA: Association for Computational Linguistics, pp. 319–326.

Ballesteros, Miguel and Yaser Al-Onaizan (2017). "AMR Parsing using Stack-LSTMs". In: *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*. Copenhagen, Denmark: Association for Computational Linguistics, pp. 1269–1275.

Banarescu, Laura, Claire Bonial, Shu Cai, Madalina Georgescu, Kira Griffitt, Ulf Hermjakob, Kevin Knight, Philipp Koehn, Martha Palmer, and Nathan Schneider (2013). "Abstract Meaning Representation for Sembanking". In: *Proceedings of the 7th Linguistic Annotation Workshop and Interoperability with Discourse*. Sofia, Bulgaria: Association for Computational Linguistics, pp. 178–186.

Banko, Michele, Michael J. Cafarella, Stephen Soderland, Matt Broadhead, and Oren Etzioni (2007). “Open Information Extraction from the Web”. In: *Proceedings of the 20th International Joint Conference on Artifical Intelligence*. IJCAI’07. Hyderabad, India: Morgan Kaufmann Publishers Inc., pp. 2670–2676.

Barzdins, Guntis and Didzis Gosko (2016a). “RIGA at SemEval-2016 Task 8: Impact of Smatch Extensions and Character-Level Neural Translation on AMR Parsing Accuracy”. In: *Proceedings of the 10th International Workshop on Semantic Evaluation (SemEval-2016)*. San Diego, California: Association for Computational Linguistics, pp. 1143–1147.

Barzdins, Guntis and Didzis Gosko (2016b). “RIGA at SemEval-2016 Task 8: Impact of Smatch Extensions and Character-Level Neural Translation on AMR Parsing Accuracy”. In: *Proceedings of the 10th International Workshop on Semantic Evaluation (SemEval-2016)*. San Diego, California: Association for Computational Linguistics, pp. 1143–1147.

Basile, Valerio, Johan Bos, Kilian Evang, and Noortje Venhuizen (2012). “Developing a large semantically annotated corpus”. In: *LREC 2012, Eighth International Conference on Language Resources and Evaluation*.

Bates, Madeleine, Sean Boisen, and John Makhoul (1990). “Developing an Evaluation Methodology for Spoken Language Systems”. In: *Speech and Natural Language: Proceedings of a Workshop Held at Hidden Valley, Pennsylvania, June 24-27,1990*.

Berant, Jonathan, Andrew Chou, Roy Frostig, and Percy Liang (2013). “Semantic Parsing on Freebase from Question-Answer Pairs”. In: *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*. Seattle, Washington, USA: Association for Computational Linguistics, pp. 1533–1544.

Bjerva, Johannes, Johan Bos, and Hessel Haagsma (2016). “The Meaning Factory at SemEval-2016 Task 8: Producing AMRs with Boxer”. In: *Proceedings of the 10th International Workshop on Semantic Evaluation (SemEval-2016)*. San Diego, California: Association for Computational Linguistics, pp. 1179–1184.

Bos, Johan (2008). “Wide-Coverage Semantic Analysis with Boxer”. In: *Semantics in Text Processing. STEP 2008 Conference Proceedings*. College Publications, pp. 277–286.

Bos, Johan, Stephen Clark, Mark Steedman, James R. Curran, and Julia Hockenmaier (2004). “Wide-Coverage Semantic Representations from a CCG

Parser". In: *COLING 2004: Proceedings of the 20th International Conference on Computational Linguistics*. Geneva, Switzerland: COLING, pp. 1240–1246.

Bos, Johan, Kilian Evang, Johannes Bjerva, Lasha Abzianidze, Hessel Haagsma, Rik van Noord, Pierre Ludmann, and Duc-Duy Nguyen (2017). "The Parallel Meaning Bank: Towards a Multilingual Corpus of Translations Annotated with Compositional Meaning Representations". In: *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics, EACL 2017, Valencia, Spain, April 3-7, 2017, Volume 2: Short Papers*, pp. 242–247.

Bowman, Samuel R., Jon Gauthier, Abhinav Rastogi, Raghav Gupta, Christopher D. Manning, and Christopher Potts (2016). "A Fast Unified Model for Parsing and Sentence Understanding". In: *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. Berlin, Germany: Association for Computational Linguistics, pp. 1466–1477.

Brandt, Lauritz, David Grimm, Mengfei Zhou, and Yannick Versley (2016). "ICL-HD at SemEval-2016 Task 8: Meaning Representation Parsing - Augmenting AMR Parsing with a Preposition Semantic Role Labeling Neural Network". In: *Proceedings of the 10th International Workshop on Semantic Evaluation (SemEval-2016)*. San Diego, California: Association for Computational Linguistics, pp. 1160–1166.

Buys, Jan and Phil Blunsom (2017). "Oxford at SemEval-2017 Task 9: Neural AMR Parsing with Pointer-Augmented Attention". In: *Proceedings of the 11th International Workshop on Semantic Evaluation (SemEval-2017)*. Vancouver, Canada: Association for Computational Linguistics, pp. 914–919.

Cai, Deng and Wai Lam (2019). "Core Semantic First: A Top-down Approach for AMR Parsing". In: *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and 9th International Joint Conference on Natural Language Processing*. Hong Kong, China: Association for Computational Linguistics.

Cai, Qingqing and Alexander Yates (2013). "Large-scale Semantic Parsing via Schema Matching and Lexicon Extension". In: *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. Sofia, Bulgaria: Association for Computational Linguistics, pp. 423–433.

Cai, Shu and Kevin Knight (2013). "Smatch: an Evaluation Metric for Semantic Feature Structures". In: *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*. Sofia, Bulgaria: Association for Computational Linguistics, pp. 748–752.

Carreras, Xavier and Lluís Màrquez (2005). "Introduction to the CoNLL-2005 shared task: Semantic role labeling". In: *Proceedings of the Ninth Conference on Computational Natural Language Learning*. Association for Computational Linguistics, pp. 152–164.

Chang, Pi-Chuan, Michel Galley, and Christopher D. Manning (2008). "Optimizing Chinese Word Segmentation for Machine Translation Performance". In: *Proceedings of the Third Workshop on Statistical Machine Translation*. Columbus, Ohio: Association for Computational Linguistics, pp. 224–232.

Chapelle, Olivier, Jason Weston, Léon Bottou, and Vladimir Vapnik (2001). "Vicinal risk minimization". In: *Advances in neural information processing systems*, pp. 416–422.

Chen, David L and Raymond J Mooney (2008). "Learning to sportscast: a test of grounded language acquisition". In: *Proceedings of the 25th international conference on Machine learning*, pp. 128–135.

Chiang, David, Jacob Andreas, Daniel Bauer, Karl Moritz Hermann, Bevan Jones, and Kevin Knight (2013). "Parsing Graphs with Hyperedge Replacement Grammars". In: *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. Sofia, Bulgaria: Association for Computational Linguistics, pp. 924–932.

Cho, Kyunghyun, Bart van Merrienboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio (2014). "Learning Phrase Representations using RNN Encoder–Decoder for Statistical Machine Translation". In: *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*. Doha, Qatar: Association for Computational Linguistics, pp. 1724–1734.

Choe, Do Kook and Eugene Charniak (2016). "Parsing as Language Modeling". In: *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*. Austin, Texas: Association for Computational Linguistics, pp. 2331–2336.

Choi, Jinho D. and Andrew McCallum (2013). "Transition-based Dependency Parsing with Selectional Branching". In: *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. Sofia, Bulgaria: Association for Computational Linguistics, pp. 1052–1062.

Christensen, Janara, Mausam, Stephen Soderland, and Oren Etzioni (2013). "Towards Coherent Multi-Document Summarization". In: *Proceedings of the 2013 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*. Atlanta, Georgia: Association for Computational Linguistics, pp. 1163–1173.

Christensen, Janara, Stephen Soderland, Oren Etzioni, et al. (2011). "An analysis of open information extraction based on semantic role labeling". In: *Proceedings of the sixth international conference on Knowledge capture*. ACM, pp. 113–120.

Chu, Y. J. and T. H. Liu (1965). "On the shortest arborescence of a directed graph". In: *Science Sinica* 14.

Clarke, James, Dan Goldwasser, Ming-Wei Chang, and Dan Roth (2010). "Driving Semantic Parsing from the World's Response". In: *Proceedings of the Fourteenth Conference on Computational Natural Language Learning*. Uppsala, Sweden: Association for Computational Linguistics, pp. 18–27.

Cohen, Jordan (2007). "The GALE project: A description and an update". In: *Automatic Speech Recognition & Understanding, 2007. ASRU. IEEE Workshop on*. IEEE, pp. 237–237.

Conneau, Alexis, Germán Kruszewski, Guillaume Lample, Loïc Barrault, and Marco Baroni (2018). "What you can cram into a single $&!#* vector: Probing sentence embeddings for linguistic properties". In: *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. Melbourne, Australia: Association for Computational Linguistics, pp. 2126–2136.

Copestake, Ann (2007). "Semantic composition with (robust) minimal recursion semantics". In: *Proceedings of the Workshop on Deep Linguistic Processing*. Association for Computational Linguistics, pp. 73–80.

Copestake, Ann (2009). "***Invited Talk:*** Slacker Semantics: Why Superficiality, Dependency and Avoidance of Commitment can be the Right Way to Go". In: *Proceedings of the 12th Conference of the European Chapter of the ACL (EACL 2009)*. Athens, Greece: Association for Computational Linguistics, pp. 1–9.

Copestake, Ann, Dan Flickinger, Rob Malouf, Susanne Riehemann, and Ivan Sag (1995). "Translation using Minimal Recursion Semantics". In: *In Proceedings of the Sixth International Conference on Theoretical and Methodological Issues in Machine Translation*.

Copestake, Ann, Dan Flickinger, Carl Pollard, and Ivan A Sag (2005). "Minimal recursion semantics: An introduction". In: *Research on Language and Computation* 3.2-3, pp. 281–332.

Daiber, Joachim, Max Jakob, Chris Hokamp, and Pablo N. Mendes (2013). “Improving Efficiency and Accuracy in Multilingual Entity Extraction”. In: *Proceedings of the 9th International Conference on Semantic Systems (I-Semantics)*.

Damonte, Marco, Shay B. Cohen, and Giorgio Satta (2017). “An Incremental Parser for Abstract Meaning Representation”. In: *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 1, Long Papers*. Valencia, Spain: Association for Computational Linguistics, pp. 536–546.

Defferrard, Michaël, Xavier Bresson, and Pierre Vandergheynst (2016). “Convolutional neural networks on graphs with fast localized spectral filtering”. In: *Advances in neural information processing systems*, pp. 3844–3852.

Del Corro, Luciano and Rainer Gemulla (2013). “Clausie: clause-based open information extraction”. In: *Proceedings of the 22nd international conference on World Wide Web*. ACM, pp. 355–366.

Devlin, Jacob, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova (2018). “Bert: Pre-training of deep bidirectional transformers for language understanding”. In: *arXiv preprint arXiv:1810.04805*.

Dong, Li and Mirella Lapata (2016). “Language to Logical Form with Neural Attention”. In: *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. Berlin, Germany: Association for Computational Linguistics, pp. 33–43.

Dorr, Bonnie and Nizar Habash (2002). “Interlingua Approximation: A Generation-Heavy Approach”. In: *In Proceedings of AMTA-2002*. University of Chicago Press.

Dowty, David R (1989). “On the semantic content of the notion of ‘thematic role’”. In: *Properties, types and meaning*. Springer, pp. 69–129.

Dozat, Timothy and Christopher D Manning (2016). “Deep biaffine attention for neural dependency parsing”. In: *arXiv preprint arXiv:1611.01734*.

Dozat, Timothy and Christopher D. Manning (2018). “Simpler but More Accurate Semantic Dependency Parsing”. In: *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*. Melbourne, Australia: Association for Computational Linguistics, pp. 484–490.

Du, Yantao, Fan Zhang, Xun Zhang, Weiwei Sun, and Xiaojun Wan (2015). “Peking: Building Semantic Dependency Graphs with a Hybrid Parser”.

In: *Proceedings of the 9th International Workshop on Semantic Evaluation (SemEval 2015)*. Denver, Colorado: Association for Computational Linguistics, pp. 927–931.

Dyer, Chris, Miguel Ballesteros, Wang Ling, Austin Matthews, and Noah A. Smith (2015). "Transition-Based Dependency Parsing with Stack Long Short-Term Memory". In: *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*. Beijing, China: Association for Computational Linguistics, pp. 334–343.

Edmonds, Jack (1968). "Optimum branchings". In: *Mathematics and the Decision Sciences, Part* 1.335-345, p. 26.

Evang, Kilian and Johan Bos (2016). "Cross-lingual Learning of an Open-domain Semantic Parser". In: *Proceedings of COLING 2016, the 26th International Conference on Computational Linguistics: Technical Papers*. Osaka, Japan: The COLING 2016 Organizing Committee, pp. 579–588.

Fader, Anthony, Stephen Soderland, and Oren Etzioni (2011). "Identifying Relations for Open Information Extraction". In: *Proceedings of the 2011 Conference on Empirical Methods in Natural Language Processing*. Edinburgh, Scotland, UK.: Association for Computational Linguistics, pp. 1535–1545.

Fader, Anthony, Luke Zettlemoyer, and Oren Etzioni (2014). "Open Question Answering Over Curated and Extracted Knowledge Bases". In: *KDD*.

Farkas, Donka F. (1988). "On obligatory control". In: *Linguistics and Philosophy* 11.1, pp. 27–58.

Faruqui, Manaal and Shankar Kumar (2015). "Multilingual Open Relation Extraction Using Cross-lingual Projection". In: *Proceedings of the 2015 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*. Denver, Colorado: Association for Computational Linguistics, pp. 1351–1356.

Fillmore, Charles J. (1968). *The case for case*. New York: Holt, Rinehart & Winston.

Flanigan, Jeffrey, Chris Dyer, Noah A. Smith, and Jaime Carbonell (2016). "CMU at SemEval-2016 Task 8: Graph-based AMR Parsing with Infinite Ramp Loss". In: *Proceedings of the 10th International Workshop on Semantic Evaluation (SemEval-2016)*. San Diego, California: Association for Computational Linguistics, pp. 1202–1206.

Flanigan, Jeffrey, Sam Thomson, Jaime Carbonell, Chris Dyer, and Noah A. Smith (2014). "A Discriminative Graph-Based Parser for the Abstract Meaning Representation". In: *Proceedings of the 52nd Annual Meeting of the*

Association for Computational Linguistics (Volume 1: Long Papers). Baltimore, Maryland: Association for Computational Linguistics, pp. 1426–1436.

Flickinger, Dan, Valia Kordoni, and Zhang Yi (2012). "DeepBank: A Dynamically Annotated Treebank of the Wall Street". In: *Proceedings of the 11th International Workshop on Treebanks and Linguistic Theories*. Lisbon, Portugal, pp. 85–86.

Foland, William and James H. Martin (2017). "Abstract Meaning Representation Parsing using LSTM Recurrent Neural Networks". In: *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. Vancouver, Canada: Association for Computational Linguistics, pp. 463–472.

Fung, Pascale and Benfeng Chen (2004). "BiFrameNet: Bilingual frame semantics resources construction by cross-lingual induction". In: *In Proceedings of the 20th International Conference on Computational Linguistics*, pp. 931–935.

Gal, Yarin and Zoubin Ghahramani (2016). "A theoretically grounded application of dropout in recurrent neural networks". In: *Advances in neural information processing systems*, pp. 1019–1027.

Ganchev, Kuzman, Jennifer Gillenwater, and Ben Taskar (2009). "Dependency Grammar Induction via Bitext Projection Constraints". In: *Proceedings of the Joint Conference of the 47th Annual Meeting of the ACL and the 4th International Joint Conference on Natural Language Processing of the AFNLP*. Suntec, Singapore: Association for Computational Linguistics, pp. 369–377.

Goodman, James, Andreas Vlachos, and Jason Naradowsky (2016). "UCL+Sheffield at SemEval-2016 Task 8: Imitation learning for AMR parsing with an alpha-bound". In: *Proceedings of the 10th International Workshop on Semantic Evaluation (SemEval-2016)*. San Diego, California: Association for Computational Linguistics, pp. 1167–1172.

Govindarajan, Venkata, Benjamin Van Durme, and Aaron Steven White (2019). "Decomposing Generalization: Models of Generic, Habitual, and Episodic Statements". In: *Transactions of the Association for Computational Linguistics* 7, pp. 501–517. eprint: https://doi.org/10.1162/tacl_a_00285.

Graves, Alex, Abdel-rahman Mohamed, and Geoffrey Hinton (2013). "Speech recognition with deep recurrent neural networks". In: *Acoustics, speech and signal processing (icassp), 2013 ieee international conference on*. IEEE, pp. 6645–6649.

Green Jr, Bert F, Alice K Wolf, Carol Chomsky, and Kenneth Laughery (1961). "Baseball: an automatic question-answerer". In: *Papers presented at the May*

9-11, 1961, western joint IRE-AIEE-ACM computer conference. ACM, pp. 219–224.

Groschwitz, Jonas, Matthias Lindemann, Meaghan Fowlie, Mark Johnson, and Alexander Koller (2018). "AMR dependency parsing with a typed semantic algebra". In: *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. Melbourne, Australia: Association for Computational Linguistics, pp. 1831–1841.

Gu, Jiatao, Zhengdong Lu, Hang Li, and Victor O.K. Li (2016). "Incorporating Copying Mechanism in Sequence-to-Sequence Learning". In: *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. Berlin, Germany: Association for Computational Linguistics, pp. 1631–1640.

Gulcehre, Caglar, Sungjin Ahn, Ramesh Nallapati, Bowen Zhou, and Yoshua Bengio (2016). "Pointing the Unknown Words". In: *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. Berlin, Germany: Association for Computational Linguistics, pp. 140–149.

Guo, Zhijiang and Wei Lu (2018). "Better Transition-Based AMR Parsing with a Refined Search Space". In: *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*. Brussels, Belgium: Association for Computational Linguistics, pp. 1712–1722.

Hajič, Jan, Eva Hajičová, Jarmila Panevová, Petr Sgall, Ondřej Bojar, Silvie Cinková, Eva Fučíková, Marie Mikulová, Petr Pajas, Jan Popelka, Jiří Semecký, Jana Šindlerová, Jan Štěpánek, Josef Toman, Zdeňka Urešová, and Zdeněk Žabokrtský (2012). "Announcing Prague Czech-English Dependency Treebank 2.0". In: *Proceedings of the Eighth International Conference on Language Resources and Evaluation (LREC-2012)*. Istanbul, Turkey: European Language Resources Association (ELRA), pp. 3153–3160.

He, Luheng, Mike Lewis, and Luke Zettlemoyer (2015). "Question-Answer Driven Semantic Role Labeling: Using Natural Language to Annotate Natural Language". In: *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*. Lisbon, Portugal: Association for Computational Linguistics, pp. 643–653.

Heim, Irene (1988). "The Semantics of Definite and Indefinite Noun Phrases". Ph. D. Dissertation. New York and London, p. 213.

Henderson, James, Paola Merlo, Ivan Titov, and Gabriele Musillo (2013). "Multilingual Joint Parsing of Syntactic and Semantic Dependencies with a Latent Variable Model". In: *Computational Linguistics* 39.4, pp. 949–998.

Hermann, Karl Moritz, Tomas Kocisky, Edward Grefenstette, Lasse Espeholt, Will Kay, Mustafa Suleyman, and Phil Blunsom (2015). “Teaching Machines to Read and Comprehend”. In: *Advances in Neural Information Processing Systems 28*. Ed. by C. Cortes, N. D. Lawrence, D. D. Lee, M. Sugiyama, and R. Garnett. Curran Associates, Inc., pp. 1693–1701.

Hershcovich, Daniel, Omri Abend, and Ari Rappoport (2017). “A Transition-Based Directed Acyclic Graph Parser for UCCA”. In: *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. Vancouver, Canada: Association for Computational Linguistics, pp. 1127–1138.

Hershcovich, Daniel, Omri Abend, and Ari Rappoport (2018). “Multitask Parsing Across Semantic Representations”. In: *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. Melbourne, Australia: Association for Computational Linguistics, pp. 373–385.

Hershcovich, Daniel, Zohar Aizenbud, Leshem Choshen, Elior Sulem, Ari Rappoport, and Omri Abend (2019). “SemEval-2019 Task 1: Cross-lingual Semantic Parsing with UCCA”. In: *Proceedings of the 13th International Workshop on Semantic Evaluation*. Minneapolis, Minnesota, USA: Association for Computational Linguistics, pp. 1–10.

Hobbs, Jerry R. (2003). “Discourse and Inference”. In:

Hochreiter, Sepp and Jürgen Schmidhuber (1997). “Long short-term memory”. In: *Neural computation* 9.8, pp. 1735–1780.

Huang, Liang and Kenji Sagae (2010). “Dynamic Programming for Linear-time Incremental Parsing”. In: *Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics*. ACL ’10. Uppsala, Sweden: Association for Computational Linguistics, pp. 1077–1086.

Huber, Peter J (1964). “Robust estimation of a location parameter”. In: *The annals of mathematical statistics*, pp. 73–101.

Hwang, Chung Hee and Lenhart K. Schubert (1994). “Interpreting tense, aspect and time adverbials: A compositional, unified approach”. In: *Temporal Logic*. Ed. by Dov M. Gabbay and Hans Jürgen Ohlbach. Berlin, Heidelberg: Springer Berlin Heidelberg, pp. 238–264.

Jean, Sébastien, Kyunghyun Cho, Roland Memisevic, and Yoshua Bengio (2015). “On Using Very Large Target Vocabulary for Neural Machine Translation”. In: *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural*

Language Processing (Volume 1: Long Papers). Beijing, China: Association for Computational Linguistics, pp. 1–10.

Ji, Heng (2009). "Cross-lingual Predicate Cluster Acquisition to Improve Bilingual Event Extraction by Inductive Learning". In: *Proceedings of the Workshop on Unsupervised and Minimally Supervised Learning of Lexical Semantics*. Boulder, Colorado, USA: Association for Computational Linguistics, pp. 27–35.

Ji, Heng, Joel Nothman, and Hoa Trang Dang (2016). "Overview of TAC-KBP2016 Tri-lingual EDL and Its Impact on End-to-End KBP". In: *Proceedings of the Text Analysis Conference (TAC)*.

Ji, Yangfeng, Chenhao Tan, Sebastian Martschat, Yejin Choi, and Noah A. Smith (2017). "Dynamic Entity Representations in Neural Language Models". In: *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*. Copenhagen, Denmark: Association for Computational Linguistics, pp. 1830–1839.

Jia, Robin and Percy Liang (2016). "Data Recombination for Neural Semantic Parsing". In: *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. Berlin, Germany: Association for Computational Linguistics, pp. 12–22.

Jiang, Wei, Zhenghua Li, Yu Zhang, and Min Zhang (2019). "HLT@SUDA at SemEval-2019 Task 1: UCCA Graph Parsing as Constituent Tree Parsing". In: *Proceedings of the 13th International Workshop on Semantic Evaluation*. Minneapolis, Minnesota, USA: Association for Computational Linguistics, pp. 11–15.

Kalchbrenner, Nal and Phil Blunsom (2013). "Recurrent Continuous Translation Models". In: *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*. Seattle, Washington, USA: Association for Computational Linguistics, pp. 1700–1709.

Kamp, H (1981). "A theory of truth and semantic representation, 277-322, JAG Groenendijk, TMV Janssen and MBJ Stokhof, eds". In:

Kamp, Hans and Uwe Reyle (1993). *From Discourse to Logic*. Dordrecht: Kluwer Academic Publishers.

Karpathy, Andrej and Li Fei-Fei (2015). "Deep visual-semantic alignments for generating image descriptions". In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 3128–3137.

Kim, Yoon, Yacine Jernite, David Sontag, and Alexander M. Rush (2016). "Character-aware Neural Language Models". In: *Proceedings of the Thirtieth*

AAAI Conference on Artificial Intelligence. AAAI'16. Phoenix, Arizona: AAAI Press, pp. 2741–2749.

Kingma, Diederik and Jimmy Ba (2014). “Adam: A method for stochastic optimization”. In: *arXiv preprint arXiv:1412.6980*.

Kiperwasser, Eliyahu and Yoav Goldberg (2016). “Simple and Accurate Dependency Parsing Using Bidirectional LSTM Feature Representations”. In: *Transactions of the Association for Computational Linguistics* 4, pp. 313–327.

Kipf, Thomas N and Max Welling (2016). “Semi-supervised classification with graph convolutional networks”. In: *arXiv preprint arXiv:1609.02907*.

Klein, G., Y. Kim, Y. Deng, J. Senellart, and A. M. Rush (2017). “OpenNMT: Open-Source Toolkit for Neural Machine Translation”. In: *ArXiv e-prints*. eprint: 1701.02810.

Koehn, Philipp, Hieu Hoang, Alexandra Birch, Chris Callison-Burch, Marcello Federico, Nicola Bertoldi, Brooke Cowan, Wade Shen, Christine Moran, Richard Zens, et al. (2007). “Moses: Open source toolkit for statistical machine translation”. In: *Proceedings of the 45th annual meeting of the ACL on interactive poster and demonstration sessions*. Association for Computational Linguistics, pp. 177–180.

Konstas, Ioannis, Srinivasan Iyer, Mark Yatskar, Yejin Choi, and Luke Zettlemoyer (2017). “Neural AMR: Sequence-to-Sequence Models for Parsing and Generation”. In: *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. Vancouver, Canada: Association for Computational Linguistics, pp. 146–157.

Kozhevnikov, Mikhail and Ivan Titov (2013). “Cross-lingual Transfer of Semantic Role Labeling Models”. In: *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. Sofia, Bulgaria: Association for Computational Linguistics, pp. 1190–1200.

Kwiatkowski, Tom, Luke Zettlemoyer, Sharon Goldwater, and Mark Steedman (2011). “Lexical Generalization in CCG Grammar Induction for Semantic Parsing”. In: *Proceedings of the 2011 Conference on Empirical Methods in Natural Language Processing*. Edinburgh, Scotland, UK.: Association for Computational Linguistics, pp. 1512–1523.

Lang, Joel and Mirella Lapata (2010). “Unsupervised Induction of Semantic Roles”. In: *Human Language Technologies: The 2010 Annual Conference of the North American Chapter of the Association for Computational Linguistics*. Los Angeles, California: Association for Computational Linguistics, pp. 939–947.

Lee, Kenton, Luheng He, Mike Lewis, and Luke Zettlemoyer (2017). "End-to-end Neural Coreference Resolution". In: *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*. Copenhagen, Denmark: Association for Computational Linguistics, pp. 188–197.

Levin, Beth and Malka Rappaport Hovav (2005). *Argument realization*. Cambridge University Press.

Liang, Chen, Jonathan Berant, Quoc Le, Kenneth D. Forbus, and Ni Lao (2017). "Neural Symbolic Machines: Learning Semantic Parsers on Freebase with Weak Supervision". In: *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. Vancouver, Canada: Association for Computational Linguistics, pp. 23–33.

Liang, Percy, Michael I. Jordan, and Dan Klein (2013). "Learning Dependency-Based Compositional Semantics". In: *Computational Linguistics* 39.2, pp. 389–446.

Lindemann, Matthias, Jonas Groschwitz, and Alexander Koller (2019). "Compositional Semantic Parsing across Graphbanks". In: *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*. Florence, Italy: Association for Computational Linguistics, pp. 4576–4585.

Ling, Wang, Phil Blunsom, Edward Grefenstette, Karl Moritz Hermann, Tomáš Kočiský, Fumin Wang, and Andrew Senior (2016). "Latent Predictor Networks for Code Generation". In: *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. Berlin, Germany: Association for Computational Linguistics, pp. 599–609.

Linzen, Tal, Emmanuel Dupoux, and Yoav Goldberg (2016). "Assessing the Ability of LSTMs to Learn Syntax-Sensitive Dependencies". In: *Transactions of the Association for Computational Linguistics* 4, pp. 521–535.

Liu, Yijia, Wanxiang Che, Bo Zheng, Bing Qin, and Ting Liu (2018). "An AMR Aligner Tuned by Transition-based Parser". In: *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*. Brussels, Belgium: Association for Computational Linguistics, pp. 2422–2430.

Liu, Zhengzhong, Teruko Mitamura, and Eduard Hovy (2015). "Evaluation Algorithms for Event Nugget Detection: A Pilot Study". In: *Proceedings of the 3rd Workshop on EVENTS at the NAACL-HLT*, pp. 53–57.

Lluís, Xavier, Xavier Carreras, and Lluís Màrquez (2013). "Joint Arc-factored Parsing of Syntactic and Semantic Dependencies". In: *Transactions of the Association for Computational Linguistics* 1, pp. 219–230.

Lluís, Xavier and Lluís Màrquez (2008). "A Joint Model for Parsing Syntactic and Semantic Dependencies". In: *CoNLL 2008: Proceedings of the Twelfth Conference on Computational Natural Language Learning*. Manchester, England: Coling 2008 Organizing Committee, pp. 188–192.

Luong, Minh-Thang, Hieu Pham, and Christopher D. Manning (2015). "Effective Approaches to Attention-based Neural Machine Translation". In: *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*. Lisbon, Portugal: Association for Computational Linguistics, pp. 1412–1421.

Lyu, Chunchuan and Ivan Titov (2018). "AMR Parsing as Graph Prediction with Latent Alignment". In: *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. Melbourne, Australia: Association for Computational Linguistics, pp. 397–407.

MacMahon, Matt, Brian Stankiewicz, and Benjamin Kuipers (2006). "Walk the Talk: Connecting Language, Knowledge, and Action in Route Instructions". In: *AAAI*.

Marcus, Mitchell, Grace Kim, Mary Ann Marcinkiewicz, Robert MacIntyre, Ann Bies, Mark Ferguson, Karen Katz, and Britta Schasberger (1994). "The Penn Treebank: Annotating Predicate Argument Structure". In: *Proceedings of the Workshop on Human Language Technology*. HLT '94. Plainsboro, NJ: Association for Computational Linguistics, pp. 114–119.

Marneffe, Marie-Catherine de, Timothy Dozat, Natalia Silveira, Katri Haverinen, Filip Ginter, Joakim Nivre, and Christopher D Manning (2014). "Universal Stanford dependencies: A cross-linguistic typology". In: *LREC*. Vol. 14, pp. 4585–4592.

Mausam, Michael Schmitz, Robert Bart, Stephen Soderland, and Oren Etzioni (2012). "Open Language Learning for Information Extraction". In: *Proceedings of Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning (EMNLP-CONLL)*.

May, Jonathan (2016). "SemEval-2016 Task 8: Meaning Representation Parsing". In: *Proceedings of the 10th International Workshop on Semantic Evaluation (SemEval-2016)*. San Diego, California: Association for Computational Linguistics, pp. 1063–1073.

May, Jonathan and Jay Priyadarshi (2017). "SemEval-2017 Task 9: Abstract Meaning Representation Parsing and Generation". In: *Proceedings of the 11th International Workshop on Semantic Evaluation (SemEval-2017)*. Vancouver, Canada: Association for Computational Linguistics, pp. 536–545.

McDonald, Ryan, Koby Crammer, and Fernando Pereira (2005). “Online Large-Margin Training of Dependency Parsers”. In: *Proceedings of the 43rd Annual Meeting of the Association for Computational Linguistics (ACL’05)*. Ann Arbor, Michigan: Association for Computational Linguistics, pp. 91–98.

McDonald, Ryan, Slav Petrov, and Keith Hall (2011). “Multi-Source Transfer of Delexicalized Dependency Parsers”. In: *Proceedings of the 2011 Conference on Empirical Methods in Natural Language Processing*. Edinburgh, Scotland, UK.: Association for Computational Linguistics, pp. 62–72.

Mei, Hongyuan, Sheng Zhang, Kevin Duh, and Benjamin Van Durme (2018). “Halo: Learning Semantics-Aware Representations for Cross-Lingual Information Extraction”. In: *Proceedings of the Seventh Joint Conference on Lexical and Computational Semantics*. New Orleans, Louisiana: Association for Computational Linguistics, pp. 142–147.

Merity, Stephen, Caiming Xiong, James Bradbury, and Richard Socher (2016). “Pointer sentinel mixture models”. In: *arXiv preprint arXiv:1609.07843*.

Miao, Yishu and Phil Blunsom (2016). “Language as a Latent Variable: Discrete Generative Models for Sentence Compression”. In: *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*. Austin, Texas: Association for Computational Linguistics, pp. 319–328.

Miller, George A. (1995). “WordNet: A Lexical Database for English”. In: *Commun. ACM* 38.11, pp. 39–41.

Mitamura, Teruko, Eric H. Nyberg, and Jaime G. Carbonell (1991). “An Efficient Interlingua Translation System for Multi-lingual Document Production”. In: *Proceedings of Machine Translation Summit III*, pp. 2–4.

Miyao, Yusuke and Jun’ichi Tsujii (2004). “Deep Linguistic Analysis for the Accurate Identification of Predicate-Argument Relations”. In: *Proceedings of Coling 2004*. Geneva, Switzerland: COLING, pp. 1392–1398.

Montague, Richard (1970). “Universal grammar”. In: *Theoria* 36.3, pp. 373–398.

Mooney, Raymond J (2014). “Semantic parsing: Past, present, and future”. In: *Association for Computational Linguistics (ACL) Workshop on Semantic Parsing*. Baltimore, Maryland.

Nallapati, Ramesh, Bowen Zhou, Cicero dos Santos, Caglar Gulcehre, and Bing Xiang (2016). “Abstractive Text Summarization using Sequence-to-sequence RNNs and Beyond”. In: *Proceedings of The 20th SIGNLL Conference on Computational Natural Language Learning*. Berlin, Germany: Association for Computational Linguistics, pp. 280–290.

Naseem, Tahira, Regina Barzilay, and Amir Globerson (2012). “Selective Sharing for Multilingual Dependency Parsing”. In: *Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. Jeju Island, Korea: Association for Computational Linguistics, pp. 629–637.

Naseem, Tahira, Abhishek Shah, Hui Wan, Radu Florian, Salim Roukos, and Miguel Ballesteros (2019). “Rewarding Smatch: Transition-Based AMR Parsing with Reinforcement Learning”. In: *arXiv preprint arXiv:1905.13370*.

Nivre, Joakim (2004). “Incrementality in Deterministic Dependency Parsing”. In: *Proceedings of the Workshop on Incremental Parsing: Bringing Engineering and Cognition Together*. IncrementParsing ’04. Barcelona, Spain: Association for Computational Linguistics, pp. 50–57.

Nivre, Joakim, Marie-Catherine de Marneffe, Filip Ginter, Yoav Goldberg, Jan Hajic, Christopher D. Manning, Ryan McDonald, Slav Petrov, Sampo Pyysalo, Natalia Silveira, Reut Tsarfaty, and Daniel Zeman (2016). “Universal Dependencies v1: A Multilingual Treebank Collection”. In: *Proceedings of the Tenth International Conference on Language Resources and Evaluation (LREC 2016)*.

Noord, Rik van and Johan Bos (2017a). “Dealing with Co-reference in Neural Semantic Parsing”. In: *Proceedings of the 2nd Workshop on Semantic Deep Learning (SemDeep-2)*. Montpellier, France: Association for Computational Linguistics, pp. 41–49.

Noord, Rik van and Johan Bos (2017b). “Neural Semantic Parsing by Character-based Translation: Experiments with Abstract Meaning Representations”. In: *Computational Linguistics in the Netherlands Journal* 7, pp. 93–108.

Oepen, Stephan, Omri Abend, Jan Hajic, Daniel Hershcovich, Marco Kuhlmann, Tim O’Gorman, Nianwen Xue, Jayeol Chun, Milan Straka, and Zdenka Uresova (2019). “MRP 2019: Cross-Framework Meaning Representation Parsing”. In: *Proceedings of the Shared Task on Cross-Framework Meaning Representation Parsing at the 2019 Conference on Natural Language Learning*. Hong Kong: Association for Computational Linguistics, pp. 1–27.

Oepen, Stephan, Marco Kuhlmann, Yusuke Miyao, Daniel Zeman, Silvie Cinkova, Dan Flickinger, Jan Hajic, and Zdenka Uresova (2015). “SemEval 2015 Task 18: Broad-Coverage Semantic Dependency Parsing”. In: *Proceedings of the 9th International Workshop on Semantic Evaluation (SemEval 2015)*. Denver, Colorado: Association for Computational Linguistics, pp. 915–926.

Oepen, Stephan, Marco Kuhlmann, Yusuke Miyao, Daniel Zeman, Dan Flickinger, Jan Hajic, Angelina Ivanova, and Yi Zhang (2014). “SemEval 2014 Task

8: Broad-Coverage Semantic Dependency Parsing". In: *Proceedings of the 8th International Workshop on Semantic Evaluation (SemEval 2014)*. Dublin, Ireland: Association for Computational Linguistics, pp. 63–72.

Oepen, Stephan and Jan Tore Lønning (2006). "Discriminant-Based MRS Banking". In: *Proceedings of the Fifth International Conference on Language Resources and Evaluation (LREC'06)*. Genoa, Italy: European Language Resources Association (ELRA).

Padó, Sebastian and Mirella Lapata (2005). "Cross-linguistic Projection of Role-semantic Information". In: *Proceedings of the Conference on Human Language Technology and Empirical Methods in Natural Language Processing*. HLT '05. Vancouver, British Columbia, Canada: Association for Computational Linguistics, pp. 859–866.

Padó, Sebastian and Mirella Lapata (2009). "Cross-lingual Annotation Projection of Semantic Roles". In: *J. Artif. Int. Res.* 36.1, pp. 307–340.

Palmer, Martha, Daniel Gildea, and Paul Kingsbury (2005). "The proposition bank: An annotated corpus of semantic roles". In: *Computational linguistics* 31.1, pp. 71–106.

Papineni, Kishore, Salim Roukos, Todd Ward, and Wei-Jing Zhu (2002). "BLEU: a method for automatic evaluation of machine translation". In: *Proceedings of the 40th annual meeting on association for computational linguistics*. Association for Computational Linguistics, pp. 311–318.

Parsons, Terence (1990). *Events in the Semantics of English*. Vol. 5. Cambridge, Ma: MIT Press.

Parton, Kristen, Kathleen R. McKeown, Bob Coyne, Mona T. Diab, Ralph Grishman, Dilek Hakkani-Tür, Mary Harper, Heng Ji, Wei Yun Ma, Adam Meyers, Sara Stolbach, Ang Sun, Gokhan Tur, Wei Xu, and Sibel Yaman (2009). "Who, What, When, Where, Why? Comparing Multiple Approaches to the Cross-Lingual 5W Task". In: *Proceedings of the Joint Conference of the 47th Annual Meeting of the ACL and the 4th International Joint Conference on Natural Language Processing of the AFNLP*. Suntec, Singapore: Association for Computational Linguistics, pp. 423–431.

Pascanu, Razvan, Tomas Mikolov, and Yoshua Bengio (2013). "On the difficulty of training recurrent neural networks". In: *Proceedings of The 30th International Conference on Machine Learning*, pp. 1310–1318.

Peng, Hao, Sam Thomson, and Noah A. Smith (2017a). "Deep Multitask Learning for Semantic Dependency Parsing". In: *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long*

Papers). Vancouver, Canada: Association for Computational Linguistics, pp. 2037–2048.

Peng, Hao, Sam Thomson, Swabha Swayamdipta, and Noah A. Smith (2018). “Learning Joint Semantic Parsers from Disjoint Data”. In: *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*. New Orle/ans, Louisiana: Association for Computational Linguistics, pp. 1492–1502.

Peng, Xiaochang, Linfeng Song, and Daniel Gildea (2015). “A Synchronous Hyperedge Replacement Grammar based approach for AMR parsing”. In: *Proceedings of the Nineteenth Conference on Computational Natural Language Learning*. Beijing, China: Association for Computational Linguistics, pp. 32–41.

Peng, Xiaochang, Chuan Wang, Daniel Gildea, and Nianwen Xue (2017b). “Addressing the Data Sparsity Issue in Neural AMR Parsing”. In: *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 1, Long Papers*. Valencia, Spain: Association for Computational Linguistics, pp. 366–375.

Pennington, Jeffrey, Richard Socher, and Christopher D. Manning (2014). “GloVe: Global Vectors for Word Representation”. In: *Empirical Methods in Natural Language Processing (EMNLP)*, pp. 1532–1543.

Plas, Lonneke van der, Marianna Apidianaki, and Chenhua Chen (2014). “Global Methods for Cross-lingual Semantic Role and Predicate Labelling”. In: *Proceedings of COLING 2014, the 25th International Conference on Computational Linguistics: Technical Papers*. Dublin, Ireland: Dublin City University and Association for Computational Linguistics, pp. 1279–1290.

Poon, Hoifung and Pedro Domingos (2009). “Unsupervised Semantic Parsing”. In: *Proceedings of the 2009 Conference on Empirical Methods in Natural Language Processing*. Singapore: Association for Computational Linguistics, pp. 1–10.

Pourdamghani, Nima, Yang Gao, Ulf Hermjakob, and Kevin Knight (2014). “Aligning English Strings with Abstract Meaning Representation Graphs”. In: *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*. Doha, Qatar: Association for Computational Linguistics, pp. 425–429.

Pradhan, Sameer, Xiaoqiang Luo, Marta Recasens, Eduard Hovy, Vincent Ng, and Michael Strube (2014). “Scoring Coreference Partitions of Predicted Mentions: A Reference Implementation”. In: *Proceedings of the 52nd Annual*

Meeting of the Association for Computational Linguistics (Volume 2: Short Papers). Baltimore, Maryland: Association for Computational Linguistics, pp. 30–35.

Price, P. J. (1990). “Evaluation of Spoken Language Systems: the ATIS Domain”. In: *Speech and Natural Language: Proceedings of a Workshop Held at Hidden Valley, Pennsylvania, June 24-27,1990*.

Pust, Michael, Ulf Hermjakob, Kevin Knight, Daniel Marcu, and Jonathan May (2015). “Parsing English into Abstract Meaning Representation Using Syntax-Based Machine Translation”. In: *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*. Lisbon, Portugal: Association for Computational Linguistics, pp. 1143–1154.

Puzikov, Yevgeniy, Daisuke Kawahara, and Sadao Kurohashi (2016). “M2L at SemEval-2016 Task 8: AMR Parsing with Neural Networks”. In: *Proceedings of the 10th International Workshop on Semantic Evaluation (SemEval-2016)*. San Diego, California: Association for Computational Linguistics, pp. 1154–1159.

Quirk, Chris, Raymond Mooney, and Michel Galley (2015). “Language to Code: Learning Semantic Parsers for If-This-Then-That Recipes”. In: *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*. Beijing, China: Association for Computational Linguistics, pp. 878–888.

Reddy, Siva, Oscar Täckström, Slav Petrov, Mark Steedman, and Mirella Lapata (2017). “Universal Semantic Parsing”. In: *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*. Copenhagen, Denmark: Association for Computational Linguistics, pp. 89–101.

Reisinger, Drew, Rachel Rudinger, Francis Ferraro, Craig Harman, Kyle Rawlins, and Benjamin Van Durme (2015). “Semantic Proto-Roles”. In: *Transactions of the Association for Computational Linguistics* 3, pp. 475–488.

Rudinger, Rachel, Adam Teichert, Ryan Culkin, Sheng Zhang, and Benjamin Van Durme (2018a). “Neural Davidsonian Semantic Proto-role Labeling”. In: *arXiv preprint arXiv:1804.07976*.

Rudinger, Rachel, Aaron Steven White, and Benjamin Van Durme (2018b). “Neural Models of Factuality”. In: *Proceedings of the Annual Meeting of the North American Association of Computational Linguistics (NAACL)*.

Rush, Alexander M., Sumit Chopra, and Jason Weston (2015). “A Neural Attention Model for Abstractive Sentence Summarization”. In: *Proceedings* *of the 2015 Conference on Empirical Methods in Natural Language Processing*. Lisbon, Portugal: Association for Computational Linguistics, pp. 379–389.

Saurí, Roser and James Pustejovsky (2009). "FactBank: a corpus annotated with event factuality". In: *Language Resources and Evaluation* 43.3, p. 227.

Schubert, Lenhart (2014). "From Treebank Parses to Episodic Logic and Commonsense Inference". In: *Proceedings of the ACL 2014 Workshop on Semantic Parsing*. Baltimore, MD: Association for Computational Linguistics, pp. 55–60.

Schubert, Lenhart (2015). "Semantic Representation". In: *Proceedings of AAAI Conference on Artificial Intelligence*.

Schubert, Lenhart K (2000). "The situations we talk about". In: *Logic-based artificial intelligence*. Springer, pp. 407–439.

Schubert, Lenhart K. and Chung Hee Hwang (2000). "Natural Language Processing and Knowledge Representation". In: ed. by Lucja M. Iwańska and Stuart C. Shapiro. Cambridge, MA, USA: MIT Press. Chap. Episodic Logic Meets Little Red Riding Hood: A Comprehensive Natural Representation for Language Understanding, pp. 111–174.

Schuster, Mike and Kuldip K Paliwal (1997). "Bidirectional recurrent neural networks". In: *IEEE Transactions on Signal Processing* 45.11, pp. 2673–2681.

See, Abigail, Peter J. Liu, and Christopher D. Manning (2017). "Get To The Point: Summarization with Pointer-Generator Networks". In: *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. Vancouver, Canada: Association for Computational Linguistics, pp. 1073–1083.

Silveira, Natalia, Timothy Dozat, Marie-Catherine de Marneffe, Samuel Bowman, Miriam Connor, John Bauer, and Christopher D. Manning (2014). "A Gold Standard Dependency Corpus for English". In: *Proceedings of the Ninth International Conference on Language Resources and Evaluation (LREC-2014)*.

Simard, Patrice Y, Yann A Le Cun, John S Denker, and Bernard Victorri (2000). "Transformation invariance in pattern recognition: Tangent distance and propagation". In: *International Journal of Imaging Systems and Technology* 11.3, pp. 181–197.

Simonyan, Karen and Andrew Zisserman (2015). "Very deep convolutional networks for large-scale image recognition". In: *International Conference on Learning Representations*.

Snover, Matthew, Xiang Li, Wen-Pin Lin, Zheng Chen, Suzanne Tamang, Mingmin Ge, Adam Lee, Qi Li, Hao Li, Sam Anzaroot, and Heng Ji (2011). "Cross-lingual Slot Filling from Comparable Corpora". In: *Proceedings of the*

4th Workshop on Building and Using Comparable Corpora: Comparable Corpora and the Web. BUCC '11. Portland, Oregon: Association for Computational Linguistics, pp. 110–119.

Srivastava, Nitish, Geoffrey E Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov (2014). "Dropout: a simple way to prevent neural networks from overfitting." In: *Journal of Machine Learning Research* 15.1, pp. 1929–1958.

Stanovsky, Gabriel and Ido Dagan (2016). "Creating a Large Benchmark for Open Information Extraction". In: *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*. Austin, Texas: Association for Computational Linguistics, pp. 2300–2305.

Stanovsky, Gabriel and Ido Dagan (2018). "Semantics as a Foreign Language". In: *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*. Brussels, Belgium: Association for Computational Linguistics, pp. 2412–2421.

Steedman, M. (1996). *Surface Structure and Interpretation*. Linguistic inquiry monographs. MIT Press.

Steedman, Mark (2000). *The Syntactic Process*. Cambridge, MA, USA: MIT Press.

Stengel-Eskin, Elias, Aaron Steven White, Sheng Zhang, and Benjamin Van Durme (2019). "Transductive Parsing for Universal Decompositional Semantics". In: *arXiv preprint arXiv:1910.10138*.

Strassel, Stephanie and Jennifer Tracey (2016). "LORELEI Language Packs: Data, Tools, and Resources for Technology Development in Low Resource Languages". In: *Proceedings of the Tenth International Conference on Language Resources and Evaluation (LREC 2016)*. Ed. by Nicoletta Calzolari (Conference Chair), Khalid Choukri, Thierry Declerck, Sara Goggi, Marko Grobelnik, Bente Maegaard, Joseph Mariani, Helene Mazo, Asuncion Moreno, Jan Odijk, and Stelios Piperidis. Portorož, Slovenia: European Language Resources Association (ELRA).

Suchanek, Fabian (2014). "Information extraction for ontology learning". In: *Lehmann and Völker [2 6]*, pp. 135–151.

Sudo, Kiyoshi, Satoshi Sekine, and Ralph Grishman (2004). "Cross-lingual information extraction system evaluation". In: *Proceedings of the 20th international Conference on Computational Linguistics*. Association for Computational Linguistics, p. 882.

Sutskever, Ilya, Oriol Vinyals, and Quoc V Le (2014). "Sequence to sequence learning with neural networks". In: *Advances in neural information processing systems*, pp. 3104–3112.

Swayamdipta, Swabha, Miguel Ballesteros, Chris Dyer, and Noah A. Smith (2016). "Greedy, Joint Syntactic-Semantic Parsing with Stack LSTMs". In: *Proceedings of The 20th SIGNLL Conference on Computational Natural Language Learning*. Berlin, Germany: Association for Computational Linguistics, pp. 187–197.

Swayamdipta, Swabha, Sam Thomson, Kenton Lee, Luke Zettlemoyer, Chris Dyer, and Noah A. Smith (2018). "Syntactic Scaffolds for Semantic Structures". In: *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*. Brussels, Belgium: Association for Computational Linguistics, pp. 3772–3782.

Szegedy, Christian, Vincent Vanhoucke, Sergey Ioffe, Jon Shlens, and Zbigniew Wojna (2016). "Rethinking the inception architecture for computer vision". In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 2818–2826.

Tang, Lappoon R and Raymond J Mooney (2001). "Using multiple clause constructors in inductive logic programming for semantic parsing". In: *European Conference on Machine Learning*. Springer, pp. 466–477.

Tenney, Ian, Patrick Xia, Berlin Chen, Alex Wang, Adam Poliak, R Thomas McCoy, Najoung Kim, Benjamin Van Durme, Sam Bowman, Dipanjan Das, and Ellie Pavlick (2019). "What do you learn from context? Probing for sentence structure in contextualized word representations". In: *International Conference on Learning Representations*.

Toutanova, Kristina, Christopher Manning, Stuart Shieber, Dan Flickinger, and Stephan Oepen (2002). "Parse disambiguation for a rich HPSG grammar". In: *First Workshop on Treebanks and Linguistic Theories (TLT2002), 253-263*. Stanford InfoLab.

Tu, Zhaopeng, Yang Liu, Zhengdong Lu, Xiaohua Liu, and Hang Li (2016). "Context gates for neural machine translation". In: *arXiv preprint arXiv:1608.06043*.

Van der Maaten, Laurens, Minmin Chen, Stephen Tyree, and Kilian Weinberger (2013). "Learning with Marginalized Corrupted Features". In: *Proceedings of The 30th International Conference on Machine Learning*.

Vashishtha, Siddharth, Benjamin Van Durme, and Aaron Steven White (2019). "Fine-Grained Temporal Relation Extraction". In: *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*. Florence, Italy: Association for Computational Linguistics, pp. 2906–2919.

Vinyals, Oriol, Łukasz Kaiser, Terry Koo, Slav Petrov, Ilya Sutskever, and Geoffrey Hinton (2015a). "Grammar as a foreign language". In: *Advances in Neural Information Processing Systems*, pp. 2773–2781.

Vinyals, Oriol, Alexander Toshev, Samy Bengio, and Dumitru Erhan (2015b). "Show and tell: A neural image caption generator". In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 3156–3164.

Wang, Chuan, Sameer Pradhan, Xiaoman Pan, Heng Ji, and Nianwen Xue (2016). "CAMR at SemEval-2016 Task 8: An Extended Transition-based AMR Parser". In: *Proceedings of the 10th International Workshop on Semantic Evaluation (SemEval-2016)*. San Diego, California: Association for Computational Linguistics, pp. 1173–1178.

Wang, Chuan and Nianwen Xue (2017). "Getting the Most out of AMR Parsing". In: *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*. Copenhagen, Denmark: Association for Computational Linguistics, pp. 1257–1268.

Wang, Chuan, Nianwen Xue, and Sameer Pradhan (2015). "A Transition-based Algorithm for AMR Parsing". In: *Proceedings of the 2015 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*. Denver, Colorado: Association for Computational Linguistics, pp. 366–375.

Wang, Mengqiu and Christopher D. Manning (2014). "Cross-lingual Projected Expectation Regularization for Weakly Supervised Learning". In: *Transactions of the Association of Computational Linguistics* 2, pp. 55–66.

Wang, Xinyu, Jingxian Huang, and Kewei Tu (2019). "Second-Order Semantic Dependency Parsing with End-to-End Neural Networks". In: *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*. Florence, Italy: Association for Computational Linguistics, pp. 4609–4618.

Wang, Yuxuan, Wanxiang Che, Jiang Guo, and Ting Liu (2018). "A Neural Transition-Based Approach for Semantic Dependency Graph Parsing". In: *AAAI Conference on Artificial Intelligence*.

White, Aaron Steven, Drew Reisinger, Keisuke Sakaguchi, Tim Vieira, Sheng Zhang, Rachel Rudinger, Kyle Rawlins, and Benjamin Van Durme (2016). "Universal Decompositional Semantics on Universal Dependencies". In: *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*. Austin, Texas: Association for Computational Linguistics, pp. 1713–1723.

White, Aaron Steven, Elias Stengel-Eskin, Siddharth Vashishtha, Venkata Govindarajan, Dee Ann Reisinger, Tim Vieira, Keisuke Sakaguchi, Sheng

Zhang, Francis Ferraro, Rachel Rudinger, Kyle Rawlins, and Benjamin Van Durme (2019). *The Universal Decompositional Semantics Dataset and Decomp Toolkit*. arXiv: 1909.13851 [cs.CL].

Whitehead, A.N. and B. Russell (1912). *Principia Mathematica*. Principia Mathematica v. 2. University Press.

Winograd, Terry (1972). "Understanding natural language". In: *Cognitive psychology* 3.1, pp. 1–191.

Wong, Yuk Wah and Raymond Mooney (2006). "Learning for Semantic Parsing with Statistical Machine Translation". In: *Proceedings of the Human Language Technology Conference of the NAACL, Main Conference*. New York City, USA: Association for Computational Linguistics, pp. 439–446.

Woods, William A, Ronald M Kaplan, Bonnie Nash-Webber, et al. (1972). "The lunar sciences natural language information system: Final report". In: *BBN report* 2378.

Xu, Kelvin, Jimmy Ba, Ryan Kiros, Kyunghyun Cho, Aaron C Courville, Ruslan Salakhutdinov, Richard S Zemel, and Yoshua Bengio (2015). "Show, Attend and Tell: Neural Image Caption Generation with Visual Attention." In: *ICML*. Vol. 14, pp. 77–81.

Yarowsky, David, Grace Ngai, and Richard Wicentowski (2001). "Inducing Multilingual Text Analysis Tools via Robust Projection Across Aligned Corpora". In: *Proceedings of the First International Conference on Human Language Technology Research*. HLT '01. San Diego: Association for Computational Linguistics, pp. 1–8.

Yin, Pengcheng and Graham Neubig (2017). "A Syntactic Neural Model for General-Purpose Code Generation". In: *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. Vancouver, Canada: Association for Computational Linguistics, pp. 440–450.

Zelle, John M. and Raymond J. Mooney (1996). "Learning to Parse Database Queries Using Inductive Logic Programming". In: *Proceedings of the Thirteenth National Conference on Artificial Intelligence - Volume 2*. AAAI'96. Portland, Oregon: AAAI Press, pp. 1050–1055.

Zeman, Daniel and Philip Resnik (2008). "Cross-language parser adaptation between related languages". In: *Proceedings of the IJCNLP-08 Workshop on NLP for Less Privileged Languages*.

Zettlemoyer, Luke and Michael Collins (2005). "Learning to Map Sentences to Logical Form: Structured Classification with Probabilistic Categorial Grammars". In: *Proceedings of the Twenty-First Conference on Uncertainty in*

Artificial Intelligence. UAI'05. Edinburgh, Scotland: AUAI Press, pp. 658–666.

Zhang, Hongyi, Moustapha Cisse, Yann N Dauphin, and David Lopez-Paz (2017a). "mixup: Beyond Empirical Risk Minimization". In: *arXiv preprint arXiv:1710.09412*.

Zhang, Sheng, Kevin Duh, and Benjamin Van Durme (2017b). "MT/IE: Cross-lingual Open Information Extraction with Neural Sequence-to-Sequence Models". In: *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 2, Short Papers*. Valencia, Spain: Association for Computational Linguistics, pp. 64–70.

Zhang, Sheng, Kevin Duh, and Benjamin Van Durme (2017c). "Selective Decoding for Cross-lingual Open Information Extraction". In: *Proceedings of the Eighth International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*. Taipei, Taiwan: Asian Federation of Natural Language Processing, pp. 832–842.

Zhang, Sheng, Xutai Ma, Kevin Duh, and Benjamin Van Durme (2019a). "AMR Parsing as Sequence-to-Graph Transduction". In: *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*. Florence, Italy: Association for Computational Linguistics, pp. 80–94.

Zhang, Sheng, Xutai Ma, Kevin Duh, and Benjamin Van Durme (2019b). "Broad-Coverage Semantic Parsing as Transduction". In: *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*. Hong Kong, China: Association for Computational Linguistics, pp. 3784–3796.

Zhang, Sheng, Xutai Ma, Rachel Rudinger, Kevin Duh, and Benjamin Van Durme (2018). "Cross-lingual Decompositional Semantic Parsing". In: *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*. Brussels, Belgium: Association for Computational Linguistics, pp. 1664–1675.

Zhang, Sheng, Rachel Rudinger, Kevin Duh, and Benjamin Van Durme (2017d). "Ordinal Common-sense Inference". In: *Transactions of the Association for Computational Linguistics* 5, pp. 379–395.

Zhang, Sheng, Rachel Rudinger, and Benjamin Van Durme (2017e). "An Evaluation of PredPatt and Open IE via Stage 1 Semantic Role Labeling". In: *IWCS 2017 — 12th International Conference on Computational Semantics — Short papers*.

Zhang, Xingxing, Liang Lu, and Mirella Lapata (2016). “Top-down Tree Long Short-Term Memory Networks”. In: *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*. San Diego, California: Association for Computational Linguistics, pp. 310–320.

Zhou, Junsheng, Feiyu Xu, Hans Uszkoreit, Weiguang QU, Ran Li, and Yanhui Gu (2016). “AMR Parsing with an Incremental Joint Model”. In: *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*. Austin, Texas: Association for Computational Linguistics, pp. 680–689.

Vita

Sheng Zhang was born in Hanyuan, Sichuan, China. He graduated from Beijing University of Posts and Telecommunications with a bachelor's degree in Information Engineering in 2012, and then graduated from Peking University with a master's degree in Computer Science in 2015. In Fall 2015, he entered the Ph.D. program in Computer Science at Johns Hopkins University, advised by Prof. Benjamin Van Durme and Prof. Kevin Duh. His research focuses on semantic parsing, common sense, and information extraction. He has interned at Microsoft Research in 2018, and Amazon Alexa AI in 2019. His paper received Best Paper Nomination at ACL 2019.